Polythecnical University of Catalonia (UPC)

Escola d'Enginyeria Industrial, Aeroespacial i Audiovisual de Terrassa (ESEIAAT)

Bachelor's Thesis:

Study of entry, descent and landing of a low mass system at Mars

A Thesis submitted by Oscar Toledo Farrando for the degree of Aerospace Engineer in the UPC.

Supervised by:
Enrique Ortega

Contents

Acknowledgements

First of all, I would like to thank my director, Enrique Ortega, for the amazing support he has been. With his support, his advice and specially his dedication to teach me, correct me and guide me, I had the possibility to develop a thesis which result would not have been possibly without him.

I also would like to acknowledge my family and my close university friends who have supported me through this path, giving me advice and supporting me when I struggled the most. It was a relief to count on them every time I needed it.

And finally, I would like to thank my partner Marina, whose personal and academical support have been one of the milestones of this thesis result.

List of Tables

List of Figures

Abstract

Since the Mariner 4 Martian mission, which was the first one in succeeding in the red planet, the spatial race to study Mars has experimented a huge growth. One of its challenges is to deal with the thin atmosphere of the red planet, which demands very accurate design of the entry, descent and landing (EDL) system. This thesis develops a preliminary design of the EDL system of a small payload (see (Pasolini et al., 2017)), which consists of an aeroshell system for the reentry and the hypersonic and supersonic phases. Once the capsule has reached a subsonic velocity, a parachute is used for descent in the last flight phase.

This project also includes the development of a parametric flight simulator to calculate the trajectory of the capsule. A CFD model is used to calculate the aerodynamic parameters of the hypersonic and supersonic phases. For the subsonic flight, as there is a wide variety of results in the literature, a suitable parachute model is selected from existing experimental data.

Aim

The aim of this project is the study of the reentry of a small capsule in Mars and the dynamic simulation the braking system of a generic geometry, both in the hypersonic and subsonic phase. In order to do that a parametric 3 DoF fligh simulator is developed. The required aerodynamic data will be extracted from a CFD study in the hypersonic and supersonic phases and from existing experimental data for the subsonic one.

Scope

The present project involves:

- Study of the most important aerodynamic characteristics that affect a blunt body.
- Selection of the phases of the reentry
- Selection of the geometry
- Development of a 3 DoF flight simulator.
- CFD study of the aerodynamic loads during the hypersonic phase
- Linearization of the aerodynamic coefficients
- Modelling of the parachute from existing experimental data
- Calculation of the final trajectory with the flight simulator

The following tasks are outside the scope of this project:

- Study of the atmosphere behaviour: temperature variations, wind incidence, etc.
- Study of the viscous effects in the CFD study: the case will be studied using the Euler approach of the Navier-Stokes equations
- Study of the structural design of the capsule
- Consideration of non-linear aerodynamic coefficients
- Aerodynamic study of the parachute: as this is a extensively studied field, this will be characterized using experimental existing data.

Requirements

The requirements of this project are:

- The total mass of the system must be 150 kg in the hypersonic phase and 110 kg in the subsonic phase (as the hypersonic braking system is jettisoned).
- The re-entry velocity is 5.5 km/s.
- The reentry altitude is 125 km (generally accepted as the beginning of the Martian atmosphere.
- The atmosphere gas must be CO_2, as is the more abundant in the Martian atmosphere
- The maximum landing velocity must be $25 m/s$.

Justification

Decades ago, computational studies were only achievable by large companies. Due to its high computational cost, they were expensive and lasted to long to use them for practical design purposes. Regarding the space exploration field, due to the complexities of the high speed flow the computational methods were difficult to implement. However, due to the development in computers and numerical methods, such analysis can be nowadays carried out in smaller research groups or even using only a personal computer, while achieving good results. This fact has potentiated that a lot of small companies and associations develop accurate designs for planetary exploration. Therefore, a lot of projects and investigations have been developed, making it easier for new researchers to achieve satisfactory results.

This thesis is developed due to the growing interest of the scientific community in the planetary exploration using computational tools. Moreover, as the investigation field is to land a small mass capsule at Mars, the work is aligned with a multitude of recent developments focused on the red planet.

Another relevant feature of this project is that the numerical analysis section is done using the open-source code OpenFOAM. Most of the commercial codes are non-editable, thus difficulting the development of new nummerical schemes. Nevertheless, in OpenFOAM the user can find multiple editable features and solvers, and even for the advanced ones, it is possible to develop new utilities. Finally, thanks to the surging interest in open-source code, there are numerous available test cases to validate the simulations present in this work.

Nomenclature

Angles

α	Angle of attack
β	Sideslip angle
γ	Flight path angle, defined between the inertial reference frame X axis and the wind reference frame X axis.
ϕ, ψ	Roll and yaw angle
θ, $\dot{\theta}$	Pitch angle and pitch velocity referred to an inertial body frame
θ_p	Angle between the parachute weight direction and its axis of symmetry
Co	Courant number

Acronims and Abreviations

AGARD	Advisory Group for Aerospace Research and Development
AoA	Angle of attack
CFD	Computational fluid dynamics
DGB	Disk gap band
DoF	Degrees of freedom
EDL	Entry, descent and landing
GHV	Generic hypersonic vehicle
IAD	Inflatable aerodynamic decelerators
M	Mach number
MOLA	Mars Orbital Laser Altimeter
ODE	Ordinary differential equations

PIMPLE Combination of PISO and SIMPLE abreviations

PISO Pressure Implicit with Splitting of Operators

RK Runge Kutta

SIMPLE Semi-implicit method for Pressure Linked Equations

Aerodynamic Coefficients

b Friction force coefficient

C_D Drag coefficient

C_L Lift coefficient

C_p Pressure coefficient

C_{D0} Parasite drag coefficient

$C_{D\alpha^2}$ Drag coefficient with respect α^2

$C_{D\alpha}$ Induced drag coefficient

C_{DP} Drag coefficient of the payload

C_{L0} Zero angle of attack lift coefficient

$C_{L\alpha}$ Lift coefficent slope

C_{M0} Free moment coefficient

$C_{M\alpha}$ Stability coeffcient

C_{MA} Aerodynamic moment coefficient

C_{mq} Damping coefficient

Geometrical Variables

δ_{M_2} Rate of reduction of the standoff distance.

δ_{sd} Standoff distance between the shock wave and the nose of the aircraft

η Surface ratio during parachute's inflation

$\overrightarrow{CG.CP}$ Distance between the gravity centre and the aerodynamic centre

$\overrightarrow{CG.PC}$ Distance from the gravity centre of the payload to the gravity centre of the parachute-payload system in a body reference frame

b Subsonic region after the shock wave

b_{ys} Rate of reduction of the subsonic area

CG Centre of gravity

CP Centre of pressure

S_P Payload reference surface

S_R Reference surface

V_{canopy} Displaced air volume due to the canopy

$\overline{c}$ Reference longitude (Diameter)

Indexes

$\hat{X}$ Flux of the X magnitude, resulting ρX

* Predicted value during the pressure corrections in the PISO algorithm

$_0$ Initial value

$_\infty$ Freestream value

$_B$ Body reference frame

$_f$ Flux across a face

$_s$ Stagnation value

$_{ij}$ Force in the i direction caused by a force in j direction

Other Symbols

Δ Infinitesimal increment

ϵ Experimental coefficient in the damping coefficient

$\frac{\delta C_m}{\delta \alpha}$ Stability derivative

$\frac{\delta}{\delta t}$ Derivative with respect the time

$\frac{D}{Dt}$ Total derivative

γ_{heat} Specific heats ratio

$\nabla \cdot \overrightarrow{X}$ Divergence of the vector $\overrightarrow{X}$

∇Y Gradient of the scalar Y

ap_{mass} Apparennt mass coefficent

c_v Velocity equation integration constant

c_v Volumetric heat capacity

C_x Opening force scale factor

c_z Height integration constant

H, G Finite difference of the convective fluxes of momentum and energy

J_1, J_b, J_c Ratio between the velocity and its derivative in the inflation force equation

k_{ij} Apparent mass coefficient

L_{BI} Rotation matrix from an inertial frame to a body frame

L_{BW} Rotation matrix from a wind reference frame to a body reference frame

M Mach number

Physical Variables

$\dot{u}$, $\dot{w}$ Velocity components in an body reference frame

$\dot{x}$, $\dot{z}$ Velocity components in an inertial reference frame

Γ Circulation

λ_G Geometrical porosity

$\overrightarrow{\omega_B}$ Angular velocity of a body frame with respect an inertial frame

$\overrightarrow{F_A}$ Aerodynamic force

$\overrightarrow{F_P}$ Payload drag force

$\overrightarrow{F_W}$ Weight Force

$\overrightarrow{F_{AM}}$ Apparent mass force

$\overrightarrow{V_A}$ Aerodynamic velocity in a wind frame

ϕ Generic variable in the convective term

ρ Density

$\mathbf{I}$ Inertia tensor

A, N Normal and axial projections of the aerodynamic forces

D, L	Drag and lift force
E	Entalphy of the system
e	Sensible energy
E_t	Total energy
g	Mars gravity acceleration
h	Altitude over martian surface
I_{yy}	Inertia product in the Y direction
m	Total vehicle mass
q	Pitch velocity referred to a body frame
t_1	Time of lines elongation
t_2	Time to separate primary and secondary bodies
t_i	Inflation time
U	Velocity in OpenFOAM syntax
V_d	Velocity at line extension
V_{sn}	Snatch velocity
$\widetilde{u}, \widetilde{w}$	Aerodynamic velocity components

Chapter 1

Introduction

Since the Mariner 4 Martian mission, which was the first one in succeeding in the red planet, the spatial race to study Mars has experimented a huge growth. One of its challenges is to deal with the thin atmosphere of the red planet, which demands very accurate design of the entry, descent and landing (EDL) system, even for low mass systems. This thesis will develop the typical EDL system of a small payload, which generally consists on an aeroshell for the hypersonic and supersonic phases and a parachute for the subsonic phase.

In this thesis, a 3 DoF flight simulator using MATLAB is developed. For the aerodynamic data, two aeroshell geometries will be analysed using OpenFOAM and one parachute will be modeled using existing experimental data. Finally, the results of the Entry, Descent and Landing (EDL) sequence are presented in the final chapter. Two descents, for each geometry, are considered: the first one will be a ballistic descent during the hypersonic and supersonic phase, and the second one will be a 2D approximation that will take into account lift and pitching moment contributions. For both cases, the descent during the subsonic phase using the parachute will be considered ballistic.

In the first chapter, a flight simulator is developed. The method used will be based on a typical Runge-Kutta (RK) staggered scheme that will solve the well-known Newton dynamic equations. The simulator will be divided in two parts: the first one will calculate the descent in the hypersonic and supersonic phase until the parachute is deployed. This parachute will be subsonic in order to avoid the complexities of supersonic flow. In order to select the order of the scheme, a set of analytical cases will be used to validate the simulator.

The second presents an aerodynamic study conducted to generate an aerodynamic model for the simulator. The hypersonic and supersonic phases, until the parachute deployment, will be calculated using CFD simulations. After that, the design of the parachute will be done using experimental existing data from other martian missions.

Regarding the CFD analysis, two approaches will be developed: the first one will use the axisymmetric form of the Navier-Stokes equations, as the analysed geometry is symmetrical with respect its axis. This analysis will be done at 0°, and the only considered force will be the drag. The second approach follows a 2D planar approximation, in order to take into consideration the contribution of the lift force. This assumption is acceptable because the capsule descent will be symmetric and will occur in a plane. Both cases will use an Eulerian approach of the Navier-Stokes equations, neglecting viscous effects (which in this type of analysis is acceptable as the main contribution over the aerodynamic forces will be due to the pressure). Finally, two types of capsules will be analysed: the first one will be a 66° cone with $D = 6$ m. The second one will be a 45° cone with $D = 3.5$ m. Finally, regarding the subsonic phase, an existing parachute's data will be used to design both the inflation and the steady state behaviour of the parachute of this study. From experimental design, several configurations will be considered in order to select the best geometry for this study.

The third and final chapters will present the results of the flight simulator using the aerodynamic model developed in the previous chapter. First of all, a comparison of the descents in the hypersonic and supersonic phases will be carried out in order to select the deploying altitude of the parachute. After that, several parachute drops will be studied in order to select the appropriate size that achieves a good deceleration and reaches terminal velocity in a realistic altitude. Finally, a stability analysis for both phases will be presented in order to mitigate the oscillations during the descent of the capsule.

1.1 Brief historical review

What has always attracted human beings to explore the Red Planet? For most of the people, it is the challenge or simply the thrill of conquering such planet before others. For others, it is the solution of finding a new home because of the allegedly extinction of our species. And even for other people, it is just the curiosity of discovering a planet which is surprisingly resembling to ours.

The study of Mars has grown considerably since 1965, when NASA's telescopes got a close-up picture of the red planet. Its orography is quite similar to the Earth, as it has polar caps, clouds and seasonal weather patterns, volcanoes, which encouraged the scientific community to seek for signs of life. That is why NASA's exploration strategy has followed three main steps, which are defined in Figure 1.1. The first missions followed the main interest of the Red Planet: the existence of water. After that, and because of the growth interest on seeking new habitable places, since 2007 the majority of the missions searched for habitability options.

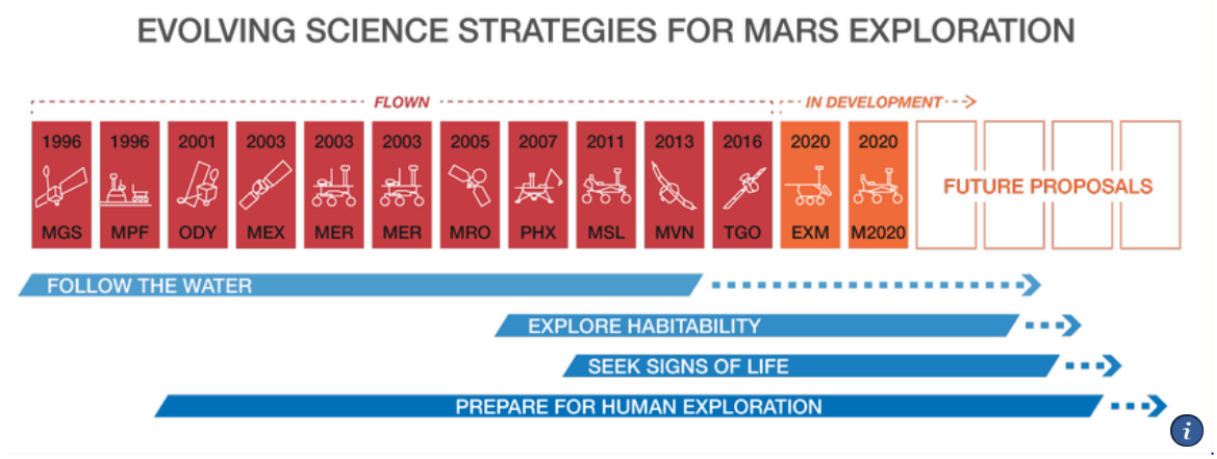


Figure 1.1: Evolution of NASA Mars missions (source (NASA))

Amongst all these missions, most of them explored the Martian surface: Viking 1 and 2, Mars Pathfinder, Mars Exploration Rovers and the Mars Science Laboratory. In order to help establish whether life ever existed in Mars, the European Space Agency created the ExoMars program in order to investigate the Martian environment and to demonstrate the possibility of creating return missions. Inside this program, a new project called "The Small Mars System" is currently being developed (Pasolini et al., 2017). It consists in delivering a small mass system onto the surface of Mars, and will carry a dust particle analyser and an aerial drone. The former should analyse the dust present in the atmosphere and the latter shall demonstrate low-altitude flight, overcoming the numerous hazards of the environment. The present work will follow the guidelines of this project, as it will also calculate the trajectory of a small mass system.

1.1.1 Typical reentry systems

One of the most complex phases of an spatial project is the reentry. And specially when Mars is the objective, the task is even tougher due to its peculiar atmosphere. In this line, the growth of numerical methods in the last decade has helped enormously ((Cruz et al., 2018), (Braun, 2010) and (Prasad and Srinivas, 2014)). During the last years, a wide range of computational studies have been done, providing accurate results of the behaviour of such stages with a much lower cost.

Regarding the reentry , as shown in (Prasad and Srinivas, 2014), there is a huge interest in analysing geometries of an Apollo shaped reentry capsules. This type of geometries are called aeroshells, which are designed to deliver payloads safely to the planet's surface. They protect the cargo from the aerodynamic heats and loads during the early reentry phases, where the flow conditions are hypersonic and supersonic. They consist of a forebody that faces the flow and which usually is a cone shaped geometry, and

an after body, which encloses the payload and has also implications in stability issues (Prasad and Srinivas, 2014).

In order to complete the descent phase of any payload, it is necessary to introduce a decelerating device with a higher drag area. For this purpose, parachutes have shown to be reliable options, specially in low-density atmospheres. These devices have shown a good decelerating behaviour at $M < 1.5$. Over this value, they have shown severe stability issues, and other options, such as inflatable toric decelerations, have proven to be better. Finally, if a parachute is going to be used, the best option is a disk-gap-band (DGB) parachute, which preformed well in a wide range of Mach numbers, even the unstable ones (Cruz et al., 2018).

Looking at future missions, several possibilities stand out. For small mass systems, two possibilities can be listed: the first one is the introduction of Computational Fluid Dynamics (CFD) methods, which will be explained in the last section. The other one is the use of inflatable aerodynamic decelerators (IADs). These devices are designed to provide deceleration and stabilization from $M = 4$ to the subsonic phase, where a safe deployment of the parachute can be done. These IADs can be packed like conventional parachutes, and are deployed directly attached to the lander. Due to their good behaviour between hypersonic and subsonic phases, they can be a perfect extra decelerating phase, between the classical aeroshells and parachutes. An example of such devices is shown in Figure 1.2

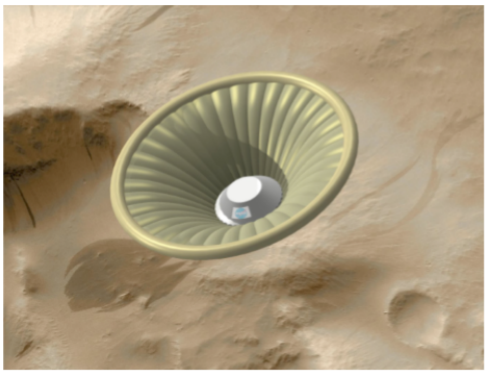


Figure 1.2: Toric cone shaped IAD

1.1.2 Some remarks on CFD simulation

As it is shown in (Tannehill and Anderson, 1997), over the past half century a surging technique for solving complex mathematical problems have risen: computational fluid dynamics. In this approach, the equations that govern the fluid flow are solved numerically. Several methods have appeared, such as finite difference methods, which were widely used at the beginning (Tannehill and Anderson, 1997), and finite volume

ones, which are one of the most used nowadays (Greenshields et al., 2009). The real revolution of this method began in the 1960s. During the past years, the designs were only available through experimental or analytical methods. With the appearence of the CFD, a third way became available. From that moment to the present days, the trend has been to migrate to these computational methods, although experiments are still needed if the flow is very complex.

This migration trend can be explained by the rapid growth of computer calculation speed. In only 40 years, the scientific community have improved the efficiency of its computers by a factor of 10^4. This tremendous technological evolution has made possible to assign CFD problems a time-gap which only 20 years ago would have been a breakthrough if the same problem had been even solved.

Several methods and algorithms have been developed during this time. It would be impossible to cite all the people who has contributed to that, therefore the methods and algorithms used for this project will be the only summarised here. First of all, the propposed CFD model will follow the finite volume method, . Inside this approach, several iterative and non iteratitve methods, such as Pressure Implicit with Splitting of Operators (PISO), will be used. Finally, these algorithms will be treated from OpenFOAM interface, an open-source group of libraries capable of dealing with numerous fluid flow and heat transfer problems (Tannehill and Anderson, 1997). The analysis in this thesis will be done using the Euler equations, thus avoiding one of the drawbacks of up-to-day CFD studies: the high computational effort to simulate properly viscous effects of the Navier-Stokes equations (see (Prasad and Srinivas, 2014) and (Viviani et al., 2012)). In this case, the key factor is the pressure gradient, which will produce the majority of the aerodynamic loads.

Chapter 2

3 DoF Flight Simulator

The aim of this section is to present the dynamic model to simulate the EDL. The methodology follows the general characteristics of the Mars project (Pasolini et al., 2017). It will be composed by two main phases:

- An initial hypersonic and supersonic reentry phase that will begin at approximately $M = 25$ and will be extended until $M = 0.75$.
- A final sub-sonic phase that will start at the end of the first phase and will last until reaching a terminal descent velocity.

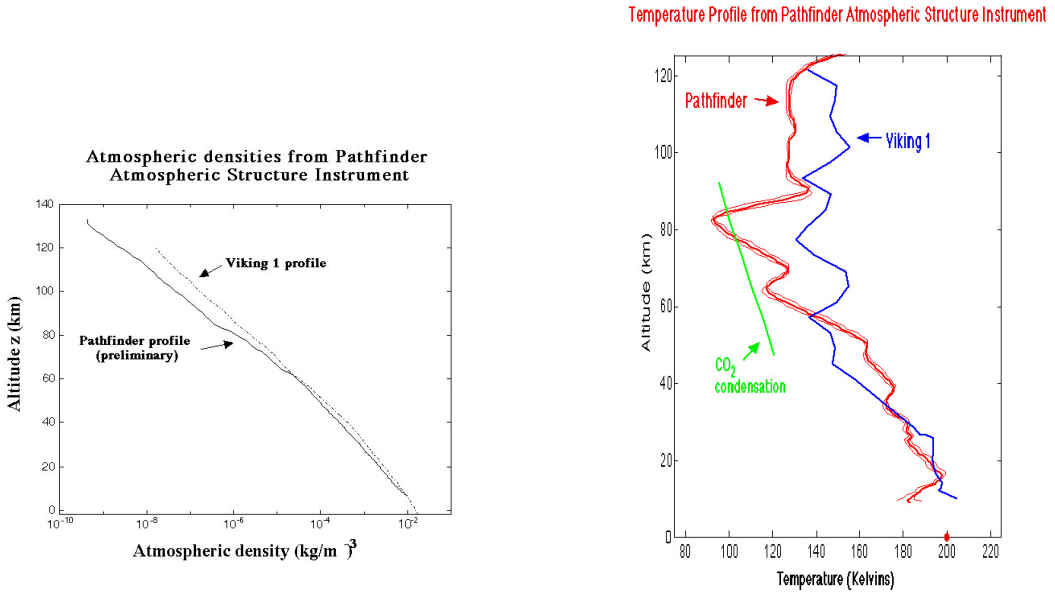
In the developing of the simulator, several suppositions have been made following (Zhang et al., 2010):

- Both the hypersonic/supersonic and subsonic decelerator systems are considered ideal rigid bodies (the parachute will be a rigid body rotating around its payload gravity centre).
- It is assumed the "flat earth" hypothesis.
- The EDL systems have no control surfaces and, moreover, they are axisymmetric bodies. This makes it acceptable the suposition that there will be no lateral forces and the descent will be cointained in a vertical plane, which encourages the developing of a 3 degree of freedom (DoF) simulator.
- Atmospheric effects like wind, dust, etc. will be neglected.

This simulator will be developed through a MATLAB code.

2.1 Atmospheric model

In order compute the atmospheric conditions at each stage of flight, density and temperature curves have been interpolated from experimental data from the Pathfinder mission. The curves are shown in Figures 2.1b and 2.1a:



(a) Density vs. altitude

(b) Temperatures vs. altitude of various missions

Figure 2.1: Atmospheric data from the Pathfinder (source: (Company))

From these figures, ρ and T have been modeled as functions of the altitude (in kilometres), as follows:

$$\rho(h) = 0.0565 \exp -0.139h \tag{1}$$

$$T(h) = \begin{cases} 3.2897h - 265.02, & \text{if } 125 < h < 120 \\ -0.1257h + 141.45, & \text{if } 120 < h < 90 \\ 4.9889h - 314.49, & \text{if } 90 < h < 80 \\ -1.5031h + 225.87, & \text{if } 80 < h < 16 \\ 1.9503h + 165.26, & \text{if } 16 < h \end{cases} \tag{2}$$

Finally, the gravity can be computed as:

$$g = g_0 \frac{R_{Mars}^2}{(R_{Mars} + h)^2} \tag{3}$$

2.2 Dynamic Model

A flight simulator developed in MATLAB is presented in this section. It will comprise the hypersonic, supersonic and subsonic phase. Despite this fact, the model for all these phases will be the same, except for some differences which will be explained in specific sections. The generic model will follow the procedure presented in (Frendreis, 2009)

and (Slegers et al., 2003).

First of all, a Newtonian approach is assumed in a body reference frame with its origin at the gravity centre of the body. The governing equations are:

$$m\dot{\overrightarrow{V_B}} + m\overrightarrow{\omega_B} \times \overrightarrow{V_B} = \sum \overrightarrow{F_B} \tag{4}$$

$$\mathbf{I}\dot{\overrightarrow{\omega_B}} + \overrightarrow{\omega_B} \times (\overrightarrow{\omega_B}\mathbf{I}) = \sum \overrightarrow{M_B} \tag{5}$$

If these equations are expanded:

$$\begin{bmatrix} \dot{u} \\ 0 \\ \dot{w} \end{bmatrix} = \frac{1}{m}\left(\overrightarrow{F_A} + \overrightarrow{F_W}\right) - \begin{bmatrix} wq \\ 0 \\ -uq \end{bmatrix} \tag{6}$$

$$\begin{bmatrix} 0 \\ \dot{q} \\ 0 \end{bmatrix} = \mathbf{I}^{-1}\left(\overrightarrow{M_A} + \overrightarrow{r_i} \times \overrightarrow{F_A}\right) \tag{7}$$

In Equation 33 the relative distance r_i is the distance between the centre of gravity and the centre of application of forces. It has been considered constant during the flight for each geometry. In addition, since the bodies are axisymmetric, this distance will be contained in the axis of symmetry.

The weight can be computed as (Tierno, 2012):

$$\overrightarrow{F_W} = L_{BI} \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} \tag{8}$$

where the components of the weight are computed in the inertial reference frame using a rotation matrix. In order to define the aerodynamic loads, the velocity components will be referred with respect a wind frame, along as the AoA. Supposing that the wind velocity is negligible in comparison with the velocity in the earth frame, then:

$$\overrightarrow{V_A} = L_{BW}^T\left(\overrightarrow{V_B} + \overrightarrow{\omega_B} \times \overrightarrow{CG.CP}\right) \tag{9}$$

$$\overrightarrow{V_A} = \begin{bmatrix} \widetilde{u} \\ 0 \\ \widetilde{w} \end{bmatrix} \tag{10}$$

$$\alpha = \arctan \frac{\widetilde{w}}{\widetilde{u}} \tag{11}$$

From this perspective, the aerodynamic forces and moments can be defined as:

$$L = \frac{1}{2}\rho S_R V_A^2 C_L \tag{12}$$

$$D = \frac{1}{2}\rho S_R V_A^2 C_D \tag{13}$$

$$M_A = \frac{1}{2}\rho\ S_R V_A^2 C_M \tag{14}$$

where the aerodynamic coefficients of Equations 13, 12 and 14 will be diferent depending on the flight phase. In the subsonic phase, the aerodynamics forces take into account more effects, such as payload drag and apparent mass. Apart from Equation 14, a pitch damping will be required due to the oscillations that are observed during the descent.

In order to obtain velocity and angular velocity in an inertial reference frame, we can use the mentioned inertial to body rotation matrix L_{BI} (Tierno, 2012):

$$\dot{x} = u\cos\theta + w\sin\theta \tag{15}$$

$$\dot{z} = -u\sin\theta + w\cos\theta \tag{16}$$

$$\dot{\theta} = q \tag{17}$$

Equations 15, 16 and 17 depend on the definition of the θ angle. In the parachute case, as the definition will be different the rotation matrix to obtain these velocities will change too.

As this is a 3 DoF simulator, the following data is imposed by the symmetry of the problem:

$$\phi = \psi = \beta = 0 \tag{18}$$

Finally, the following set of equations can be obtained:

$$\begin{bmatrix} \dot{u} \\ \ddot{\theta} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} \frac{1}{m}(-A - mg(s\theta)) - w\dot{\theta} \\ \frac{M_A}{I_{YY}} - C_{mq}\dot{\theta} \\ \frac{1}{m}(-N + mg(c\theta)) + u\dot{\theta} \end{bmatrix} \tag{19}$$

$$A = Dc\alpha - Ls\alpha \tag{20}$$

$$N = Ds\alpha + Lc\alpha \tag{21}$$

where the variable q has been substituted by $\dot{\theta}$, and the forces A and N are the result of rotating the lift and drag forces. The moment of the aerodynamic loads is computed in M_A. The algorithm is summarized in the flowchart presented in Figure 2.2:

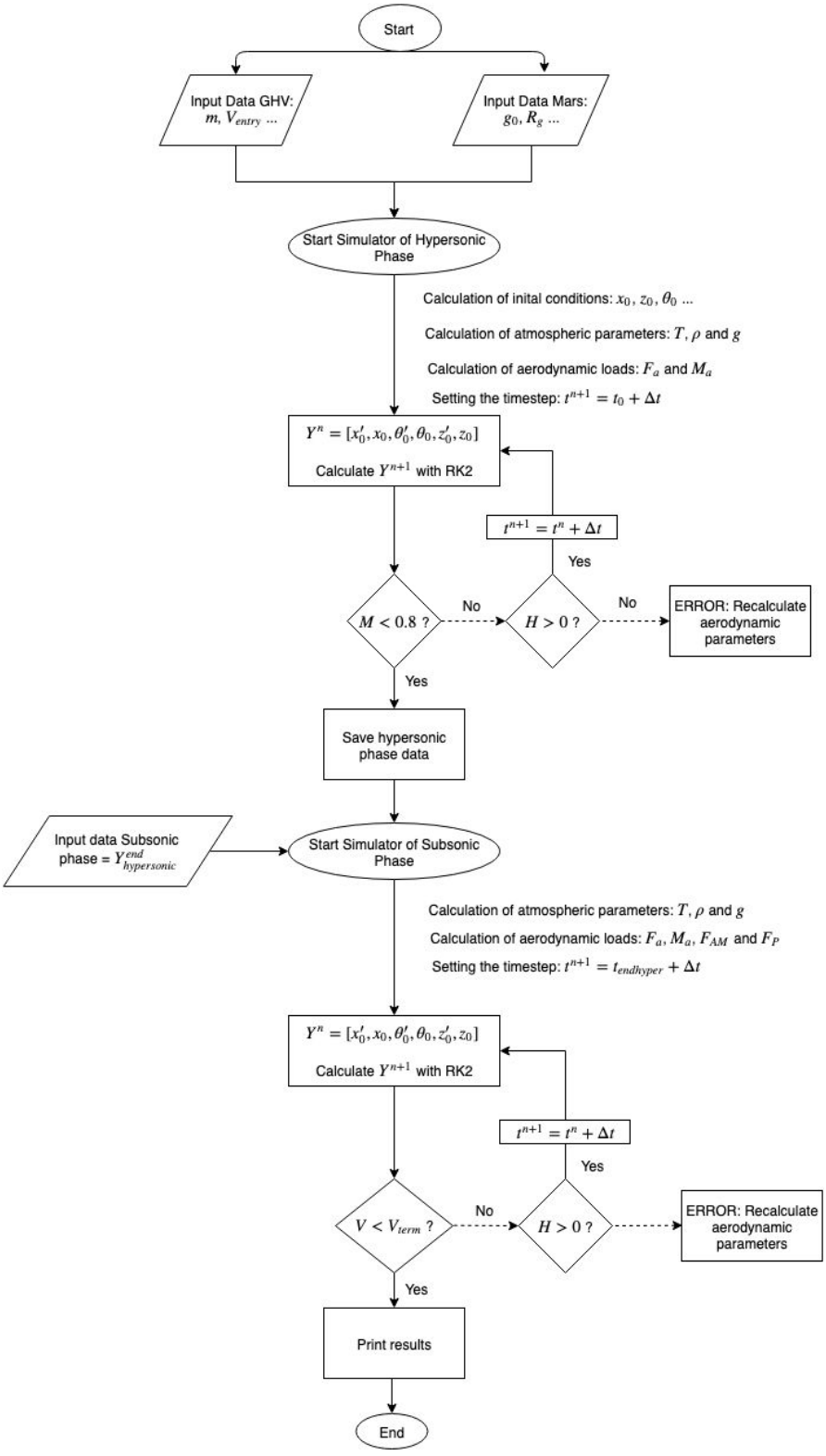


Figure 2.2: Flowchart of the flight simulator

2.2.1 Hypersonic reentry phase

The proposed model follows the assumptions given in (Chao et al., 2015). One of the best geometries for decelerating capsules is a truncated cone like that shown in Figure 2.3. This geometry generates a detached bow shock with relatively small dimensions. Moreover, due to its shape, it can always stay inside the shock wave, preventing fatal events such as high gradients in pressure and temperature.

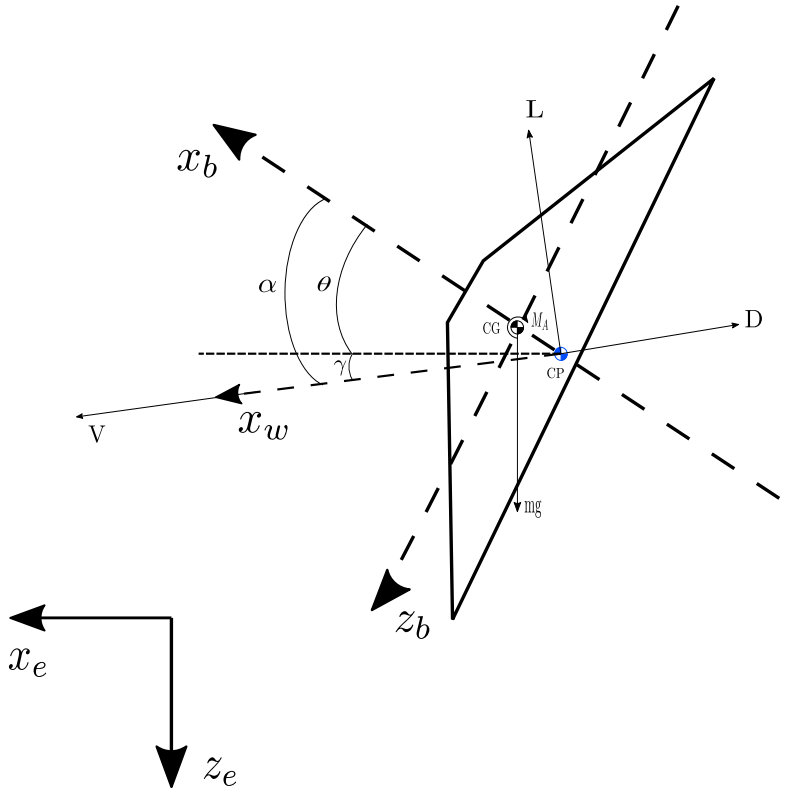


Figure 2.3: GHV forces diagram

2.2.1.1 Aerodynamic forces

The forces acting on the vehicle will be the lift and the drag, as well as the aerodynamic moment. This forces will be obtained using a 2D approximation. Moreover, another case will be considered: a ballistic descent, which will only take into account the drag force. This latter analysis will be done using an axisymetric model.

As the aerodynamic data will be obtained through a CFD simulation, the aerodynamic parameters need to be linearized. The generic hypersonic vehicle (GHV) works on a wide Mach number range (from $M = 25$ to $M = 0.75$). This means that the aerodynamic coefficients will depend on the AoA and M.

For this application, it has been found that:

$$C_L = C_{L0} + C_{L\alpha}\alpha \tag{22}$$

$$C_D = C_{D0} + C_{D\alpha}\alpha + C_{D\alpha^2}\alpha^2 \tag{23}$$

$$C_{MA} = C_{M0} + C_{M\alpha}\alpha \tag{24}$$

In order to obtain them, a linearization on alpha will be done at each Mach number, and after that a suitable polynomial will be adjusted for the Mach number dependency. Finally, is expected that $C_{L0} = C_{M0} = 0$ because the geometry is axisymmetric.

During high velocity phases, the descent is usually oscillating. Even the oscillations do not usually exceed $\pm 1°$, it is necessary to introduce a damping in the capsule because even this little oscillations would be fatal for the structure. The pitch damping coefficient will depend on the velocity, because at higher velocities the damping required is higher than at lower ones. This is mainly due to the instabilities found in transonic flow, which induces a periodic behaviour to the capsule (Moretti et al., 1987). The coefficient will then be expressed as:

$$C_{mq} = -\epsilon \frac{V_\infty}{V_{entry}}^3 \frac{1}{s} \tag{25}$$

The values in Equation 25 will be found experimentally.

2.2.1.2 Equations of motion

In the hypersonic phase, the governing equations are the following:

$$\begin{bmatrix} \dot{u} \\ \ddot{\theta} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} \frac{1}{m}(-A - mg(s\theta)) - w\dot{\theta} \\ \frac{M_A}{I_{YY}} + \frac{\sqrt{u^2+w^2}}{V_{entry}}\dot{\theta} \\ \frac{1}{m}(-N + mg(c\theta)) + u\dot{\theta} \end{bmatrix} \tag{26}$$

In the ballistic descent model, $L = 0$, and the moments acting on the capsule will be taken as $-NX_{cgcp} + C_{mq}\dot{\theta}$.

As there are 2 ODEs of first order and one of second order, four initial conditions are needed. Moreover, two more initial conditions are needed in order to compute the initial positions. The initial conditions for the hypersonic phase will be the following:

- Initial altitude, z_0: $125000m$ (beginning of the martian atmosphere).
- Inital horitzontal position x_0: $0m$. In this thesis , the important analysis is the descent, other positional effects are not important.

- Reentry flight path angle γ: -13. This value is presented in (Pasolini et al., 2017) as an optimal value to avoid a rebound to the atmosphere. Moreover, suposing $AoA_0 = 0$, $\theta_0 = 13$, as $\theta = AoA + \gamma$

- Reentry velocity V_0. $5500\frac{m}{s}$. Again, this vlaue is taken from (Pasolini et al., 2017). With this value and the one of θ_0, the velocities in the inertial body frame $\dot{x}_0$ and $\dot{z}_0$ can be calculated

- Initial pitching velocity $\dot{\theta}$: $0\frac{rad}{s}$. The body will be supposed to enter with no rotation

2.2.2 Subsonic reentry phase

The decelerating system follows the models presented in (Knacke, 1991) and (Cockrel et al., 1987). The diagram of forces is shown in Figure 2.4. The model will be supposed to descent in a ballistic trajectory, where only the effect of the weight, the drag and the apparent mass forces are taken into account. The data and dimensions will be obtained from existing experimental data.

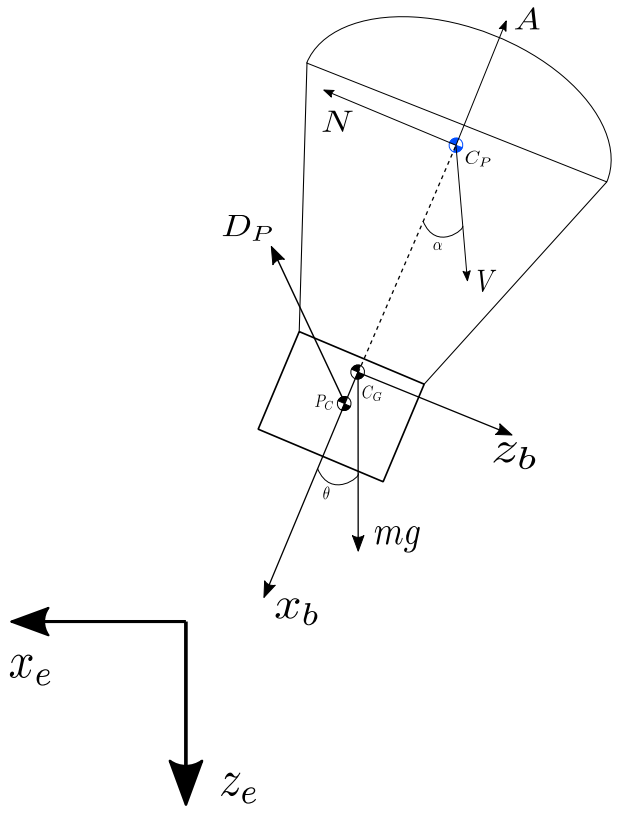


Figure 2.4: Parachute forces diagram

In this model, the stability is achieved through the contribution of each force, in this case, the drag of the payload and the canopy.

2.2.2.1 Apparent mass

According to (Lingard, 1995), when a body moves through a fluid it makes the latter to displace and therefore to exercise a force upon it. As the parachute is a low mass system, this force has important effects, such as in the dynamic stability in the pitch angle when the parachute is descending. In order to model this effect in the analysed parachute, the approach presented in (Cockrel et al., 1987) will be followed. There, a rapid inflation is supposed, because then the fluid is considered irrotational and incompressible.

In order to take into account the apparent mass, the force of the body immersed in a fluid and the fluid itself will be considered as follows:

$$F_{AM} = \frac{d}{dt}(ap_{mass}V_A) \tag{27}$$

From (Cockrel et al., 1987), the coefficient ap_{mass} can be expressed as:

$$k_{ij} = \frac{ap_{mass}}{\rho V_{canopy}} \tag{28}$$

where i indicates the direccion of the measured force and j the acceleration that caused it. Moreover, V_{canopy} will be considered to be half of an hemisphere's volume with the nominal diameter D_c, which is conventionally consiered to be $\frac{\pi D_c^2}{12}$.

In a 3 DoF conventional parachute, the only possible coefficients to be considered amongst the significant ones would be k_{11}, k_{33}, k_{55} and k_{35}. The last two coefficients can be neglected if the reference system of the parachute is near the canopy centre of pressure. As in this cases the reference system used to calculate the dynamics of the parachute is in the centre of the canopy, the only remaining coefficients are k_{11} and k_{33}. Their values for conventional parachutes can be found in (Cockrel et al., 1987), and they correspond to:

$$k_{11} = 0.7 \tag{29}$$

$$k_{33} = 0.21 \tag{30}$$

In the case of conventional parachutes, it is shown that as $k_{11} > k_{33}$, the value of k_{11} coefficient has a destabilizing effect on parachute's oscillations.

Finally, in order to simply the calculations, the application point of this forces will be considered to be in the centre of pressure of the parachute.

2.2.2.2 Aerodynamic forces

As the descent is ballistic, the only force considerd is the drag. It will be extracted from an existing parachute model of the Viking mision (see (Moog, 1973)). This drag function will take into account the inflation time of the parachute. The design procedure of the parachute is explained in Chapter 3, section 3.6.

In this case, the C_D function of the parachute will be:

$$C_D = C_{D0} \tag{31}$$

where, again, the C_D depends on the Mach (the range of Mach is lower but still the drag has noticeable changes). Finally, the pitch damping for the parachute has been experimentally determined.

2.2.3 Equations of motion

In the subsonic case, the equations of motion need to take into account the contribution of apparent mass forces and payload forces. Moreover, the pitching angle θ is defined between the weight and the x_b vector.

Recalling Equations 32 and 33:

$$\begin{bmatrix} \dot{u} \\ 0 \\ \dot{w} \end{bmatrix} = \frac{1}{m}\left(\overrightarrow{F_A} - \overrightarrow{F_W} + \overrightarrow{F_A M} + \overrightarrow{F_P}\right) - \begin{bmatrix} wq \\ 0 \\ -uq \end{bmatrix} \tag{32}$$

$$\begin{bmatrix} 0 \\ \dot{q} \\ 0 \end{bmatrix} = \mathbf{I}^{-1}\left(\overrightarrow{CG.PC} \times \overrightarrow{F_P} + \overrightarrow{CG.CP} \times \overrightarrow{F_A}\right) \tag{33}$$

Before introducing the final set of equations of the parachute, it is important to define the rotations between the body and the inertial reference fram, as the pitching angle of the parachute θ_p is defined differently. Therefore, modifying Equations 15 and 16:

$$\dot{x} = u\sin\theta_p - w\cos\theta_p \tag{34}$$

$$\dot{z} = u\cos\theta_p + w\sin\theta_p \tag{35}$$

Regarding the parachute's stability, the following consideration has been done: the parachute will be taken as a rigid body rotating on its payload gravity center. Therefore, the equilibrium will be achieved when the normal force compensates the contribution of the weight of the canopy. Hence:

$$\ddot{\theta_p} = \frac{1}{I_{YY}}(-N + m_{canopy}\sin\theta_p)(X_{cgcp} + X_{cgpc}) + C_{mq}\dot{\theta_p} \tag{36}$$

The final set of equations results:

$$\begin{bmatrix} \dot{u} \\ \ddot{\theta}_p \\ \dot{w} \end{bmatrix} = \begin{bmatrix} \frac{1}{m}(-A + mg(c\theta_p) - F_{Px} + F_{AMx}) - w\dot{\theta}_p \\ \frac{1}{I_{yy}}(-N + m_c \sin\theta_p)(X_{cgcp} + X_{cgpc}) + C_{mq}\dot{\theta}_p \\ \frac{1}{m}(-N + mg(s\theta_p) - F_{Pz} + F_{AMz}) + u\dot{\theta}_p \end{bmatrix} \tag{37}$$

As well as in the hypersonic case, initial data is needed. The initial height z_0, the initial horitzontal position x_0 and the initial velocity will be taken from the output data of the last stage of the hypersonic phase. Regarding the initial AoA, the initial pitching angle and the initial pitching velocity, they will be considered differently because of the parachute inflation:

- Initial pitching angle θ: it will be considered to range between 0 to 30°, which have shown good stability behaviour.
- Initial pitching velocity $\dot{\theta}$: $0\frac{rad}{s}$. The body will be supposed to start the subsonic descent with no rotation

2.3 Numerical Integration

In order to integrate in time the flight dynamic model, several numerical schemes have been implemented. In the next, the performance of first, second and fourth order Runge-Kutta schemes with different time steps is investigated (Shampine, 1984). The analysis is also intended to verify the proper behavior of the solution schemes.

2.3.1 Verification test cases

In order to compare the different numerical schemes, two well-known verification test cases are adopted:

- A free fall with a drag force proportional to v^2. (C.1)
- A parabolic movement, with drag force proportional to velocity and pitch damping. (C.2)

The input data supposed for each case is the following:

Cases	x_{e0} (m)	z_{e0} (m)	g $(\frac{m}{s^2})$	ρ $(\frac{kg}{m^3})$
C.1	0	1500	3,71	0,015
C.2	0	0	3,71	0,015

Table 2.1: Atmospheric data

Cases	θ_0 (°)	m (kg)	I_{yy} $(\frac{kg}{m^2})$	C_{d0}	V_0 $(\frac{m}{s})$	q_0 $(\frac{rad}{s})$	C_{mq}
C.1	-90	150	15	0,1	500	0	0
C.2	45	150	15	0,1	500	1	0,5

Table 2.2: Vehicle data

where the parameter θ_0 in Table 2.2 is used to force, respectively, a free fall with only velocity in the z axis and a parabolic motion pointing in the positive X and Z directions.

Note that although the second case is not a realistic one, it is of high interest to this application because it integrates all the ODEs that solve the variables of the problem.

2.3.1.1 Results for the free fall

Considering the following ODE:

$$m\frac{dv}{dt} = mg - bv^2 \tag{38}$$

it is possible to analytically integrate the velocity if the movement is contained in the z axis. Integrating one time, the velocity is obtained Bizen (2015):

$$v = k\frac{1 + c_v e^{\frac{-2kbt}{m}}}{1 - c_v e^{\frac{-2kbt}{m}}} \tag{39}$$

where c_v is the integration constant, $k = \sqrt{\frac{gm}{b}}$ and $b = \frac{1}{2}\rho S_{ref} C_{D0}$. Integrating again, it is possible to obtain the vertical position function:

$$z = k\left(t + \frac{2}{A}\ln\left(1 - c_v e^{-At}\right)\right) + c_z \tag{40}$$

where c_z is the integration constant and $A = \frac{2kb}{m}$.

These results have been compared with numerical integration. The convergence of the error in Figures 2.5 and 2.6 is:

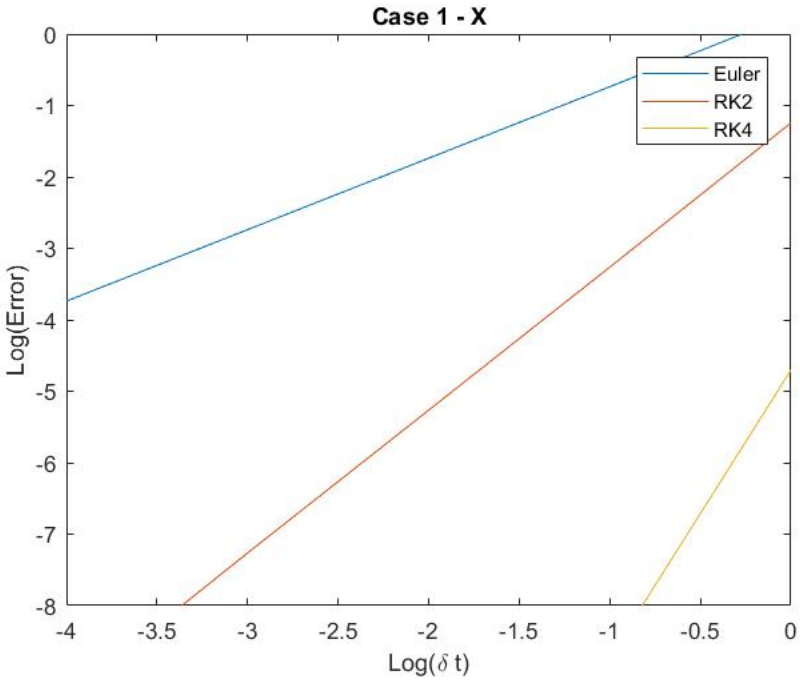


Figure 2.5: Error in X

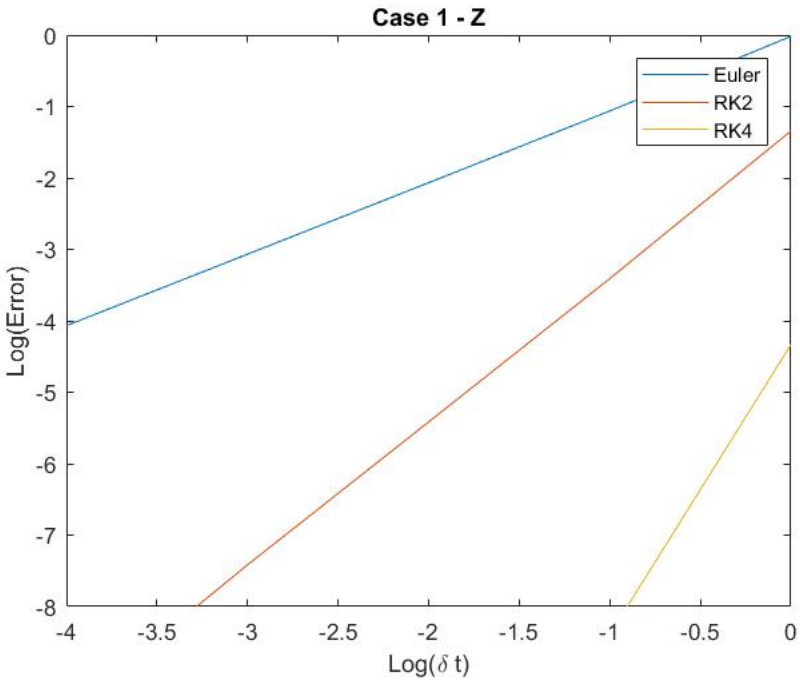


Figure 2.6: Error in Z

As it can be observed from the results, all the schemes provide an acceptable result using any time step from 10^0 to 10^{-4}. Moreover, they behave as expected because the slopes for the logaritmic plots are -1 for the Euler case, -2 for the RK2 and -4 for the RK4 scheme.

2.3.1.2 Results for the parabolic motion

In this case, the equations studied are the following:

$$m\frac{dv_x}{dt} = -bv_x \tag{41}$$

$$m\frac{d^2\theta}{dt^2} = -C_{mq}\frac{d\theta}{dt} \tag{42}$$

$$m\frac{dv_z}{dt} = -mg - bv_z \tag{43}$$

Using the basic integration procedures, it is possible to obtain:

$$x = \frac{m}{b}V_{x0}(1 - e^{\frac{bt}{m}}) \tag{44}$$

$$z = \frac{mgt}{b} + \frac{m}{b}(V_{z0} - \frac{mg}{b})(1 - e^{\frac{bt}{m}}) \tag{45}$$

$$\theta = \theta_0 + q_0\frac{I_{yy}}{C_{mq}}(1 - e^{\frac{I_{yy}}{C_{mq}}}) \tag{46}$$

Here, comparing both analytical and numerical solutions, the results are the following:

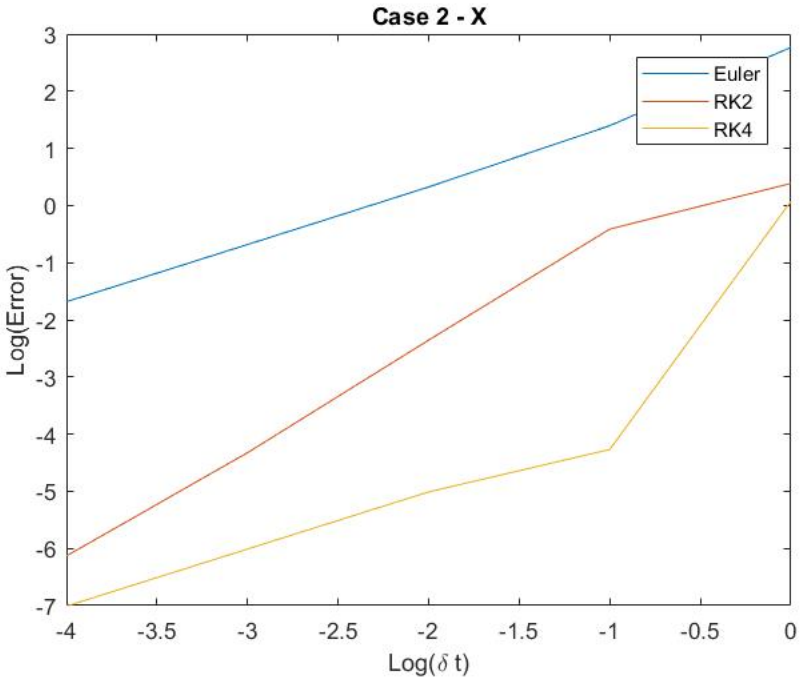


Figure 2.7: Error in X

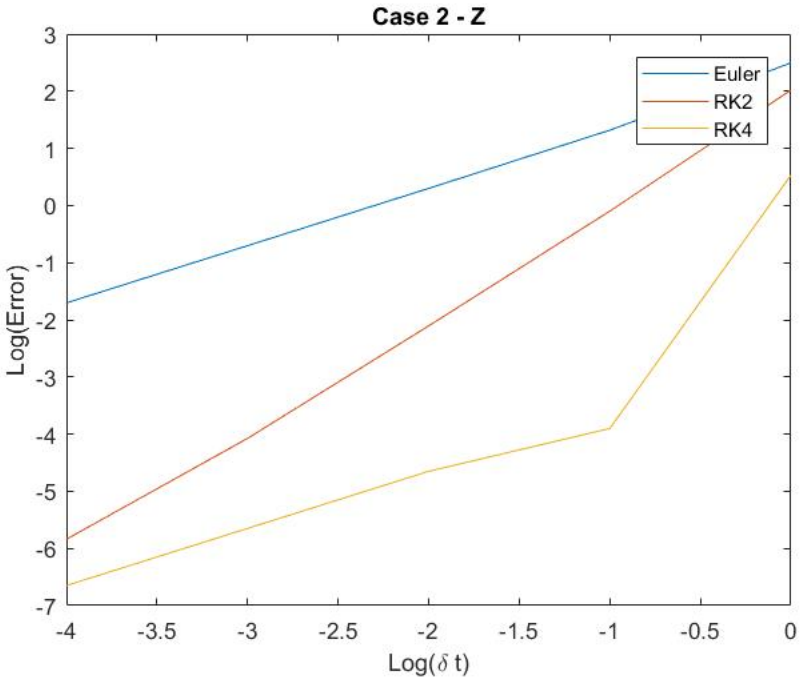


Figure 2.8: Error in Z

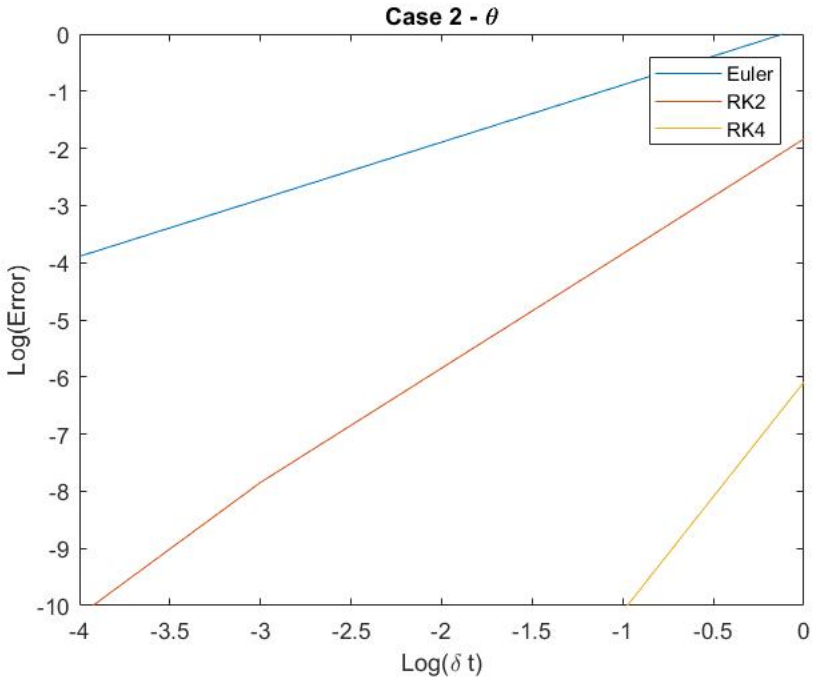


Figure 2.9: Error in θ

In this last case, the slopes of converge of both the Euler and RK schemes are again the expected. In order to save computational time but without a lose of accuracy, it has been decided to use a **RK2 scheme** with a time step of $10^{-3}s$, where the error is below $0,0001\%$ in all the analysed variables, which is acceptable. RK4 has been discarded as it achieves an accuracy less than 10^{-10}, which can induce to round-off errors.

Note that in Figure 2.7, the behaviour of the RK2 and RK4 schemes is not the expected (the same occurs in Figure 2.8 for the RK4 scheme). These results can be due to small interactions between the three solved variables, and further iterations should be performed in order to stablish why they occur. However, as the selected scheme is the RK2 one, these results have been taken as valid one because the discrepancies are very low and under a margin of acceptance.

Chapter 3

Aerodynamics Analysis

In this chapter, the aerodynamic analysis will be presented. Firstly, the governing equations will be discussed, as well as the discretizing method. Secondly, an overview of the solvers will be presented. Furthermore, a validation case will be presented to select the mesh and the solver simulation parameters for the final CFD analysis. Then, the aeroshell analysis will be presented. Finally, the design of the parachute will be explained, as well as the final aerodynamic model for the whole EDL phases.

3.1 Governing equations

The governing equations for a compressible external flow are the Navier-Stokes (NS) equations of continuity, momentum and energy, as well as the equation of state of gases (in this case, considered perfect and with a heat transfer ratio of 1.4). In this case, as the viscous effects are neglected, the governing equations will be taken in its inviscid form, i.e. the Euler approach (see (CTTC, 2019) and (Versteeg and W, 2007)).

The Euler equations consist of mass, momentum and energy conservation statements. In its vectorial form, mass conservation (continuity equation) can be written as follows (Anderson, 1995):

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{V}) = 0 \tag{47}$$

This form of the continuity equation is written in the Eulerian form, commonly used in fluid dynamics. The first term accounts for the rate of increase of the density in the control volume and the second one represents the rate of mass flux passing out the control surface. Following the continuity equation, there is the momentum conservation equation:

$$\frac{\partial (\rho V)}{\partial t} + \nabla \cdot (V(\rho V)) = -\nabla p \tag{48}$$

where in the right hand side, the first element represents the rate of increase of momentum per unit volume in the control volume and the second one the rate of momentum lost by convection through its surfaces. The last element represents the pressure forces over the faces of the control volume. Finally, the energy equation is presented:

$$\frac{\partial(\rho E)}{\partial t} + \nabla \cdot (V(\rho E)) = -\nabla \cdot (Vp) \tag{49}$$

where in Equation 49, $E = e + \frac{V^2}{2}$ and there are no heat fluxes. In this equation, the first term represents the rate of increase of total energy per unit volume. The second one represents the rate of energy lost by convection and finally, the left hand term represents the work done by the pressure forces.

To complete this system of equations, the equation of state for perfect gases is

$$P = \rho R T \tag{50}$$

and the equation of the total energy is

$$E_t = e + \frac{V^2}{2} - \frac{p}{\rho} \tag{51}$$

where the temperature can be expressed, in a perfect gas

$$T = \frac{e}{C_v} \tag{52}$$

3.2 Computational approach

The aerodynamics problem is going to be solved with OpenFOAM. In order to solve the Euler equations, the finite volume method is adopted. In this case, the FV method is applied in polyhedral cells, like the ones depicted in Figure 3.1, and which will divide the domain into a certain number of cells. These cells have faces that each one accomplishes one of the following statements: either they are internal and intersect *two cells* or they comprise a boundary. In the first case, this face will be assigned an owner cell (P in Figure 3.1) and a neighbour (denoted by N). As it can be seen, the vector S_f is normal to the face and points out the owner cell (Greenshields et al., 2009).

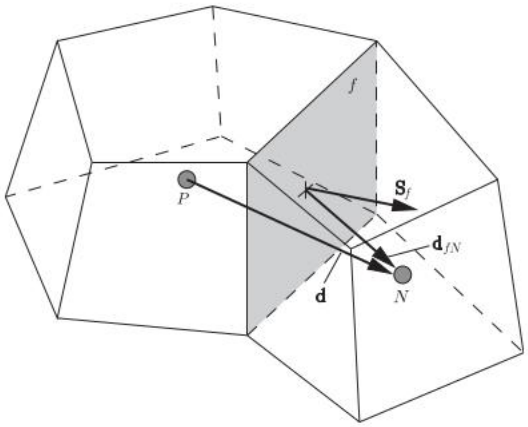


Figure 3.1: Finite volume discretization (source: (Greenshields et al., 2009))

All the variables are stored in each cell owner's centroid. Moreover, the vector d connects both centroids and d_{fN} connects the face centre with the neighbour's centroid. Finally, in order to obtain the solutions of the governing equations in each cell, they will be expressed as an integral over the control volume and discretized and converted to surface integrals using the Gauss theorem. For more information about this method, see Appendix D.

In this work, two algorithms are used. First of all, for the hypersonic and supersonic phase, the *rhoCentralFoam* will be used. This density-based solver, based in the central-upwind schemes developed by (Kurganov and Tadmor, 2000), is claimed to be one of the most widely used for these type of flows. In (Bondarev, 2018), a validation study has been done over a supersonic cone, and it has concluded that the solver with higher accuracy is the *rhoCentralFoam*.

An extensive explanation of its discretization method can be found in Appendix D.1. The following Figure shows an the flux diagram of the algorithm (Figure 3.2).

The other algorithm that is going to be used is the *sonicFoam*, a pressure based solver based on the PISO algorithm, suitable for transonic and low subsonic flow (Issa, 1981). This algorithm is going to be used for $M < 1.5$, because under low supersonic and transonic conditions, the pressure based solvers perform better than the density based ones. Despite in (Bondarev, 2018) it has a higher error than *rhoCentralFoam*, is the second one in the reccomendation list.

As well as in the *rhoCentralFoam* algorithm, an extensive explanation can be found in Appendix D.2. In Figure 3.3 it can be consulted its flowchart:

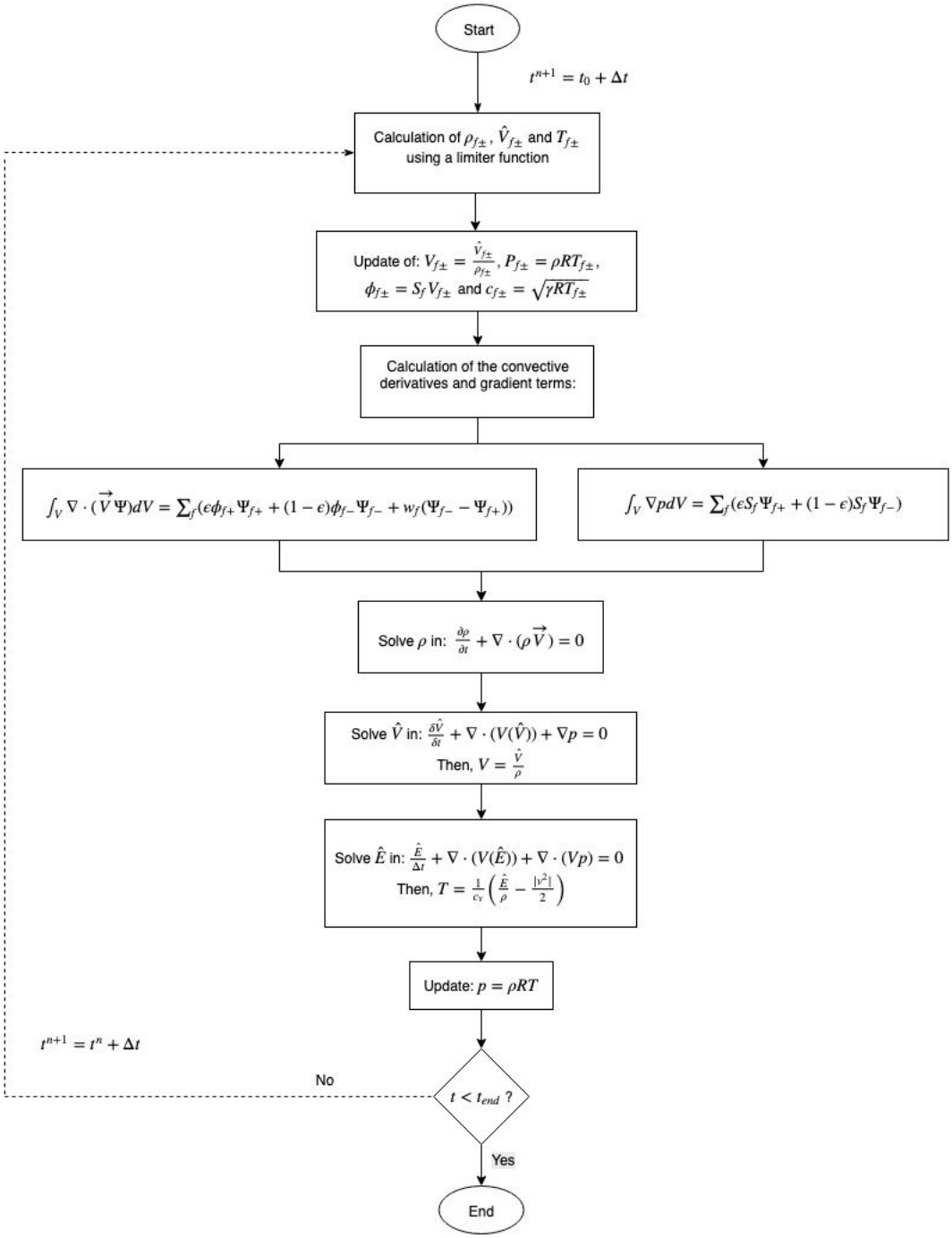


Figure 3.2: Flowchart of *rhoCentralFoam* resolution algorithm

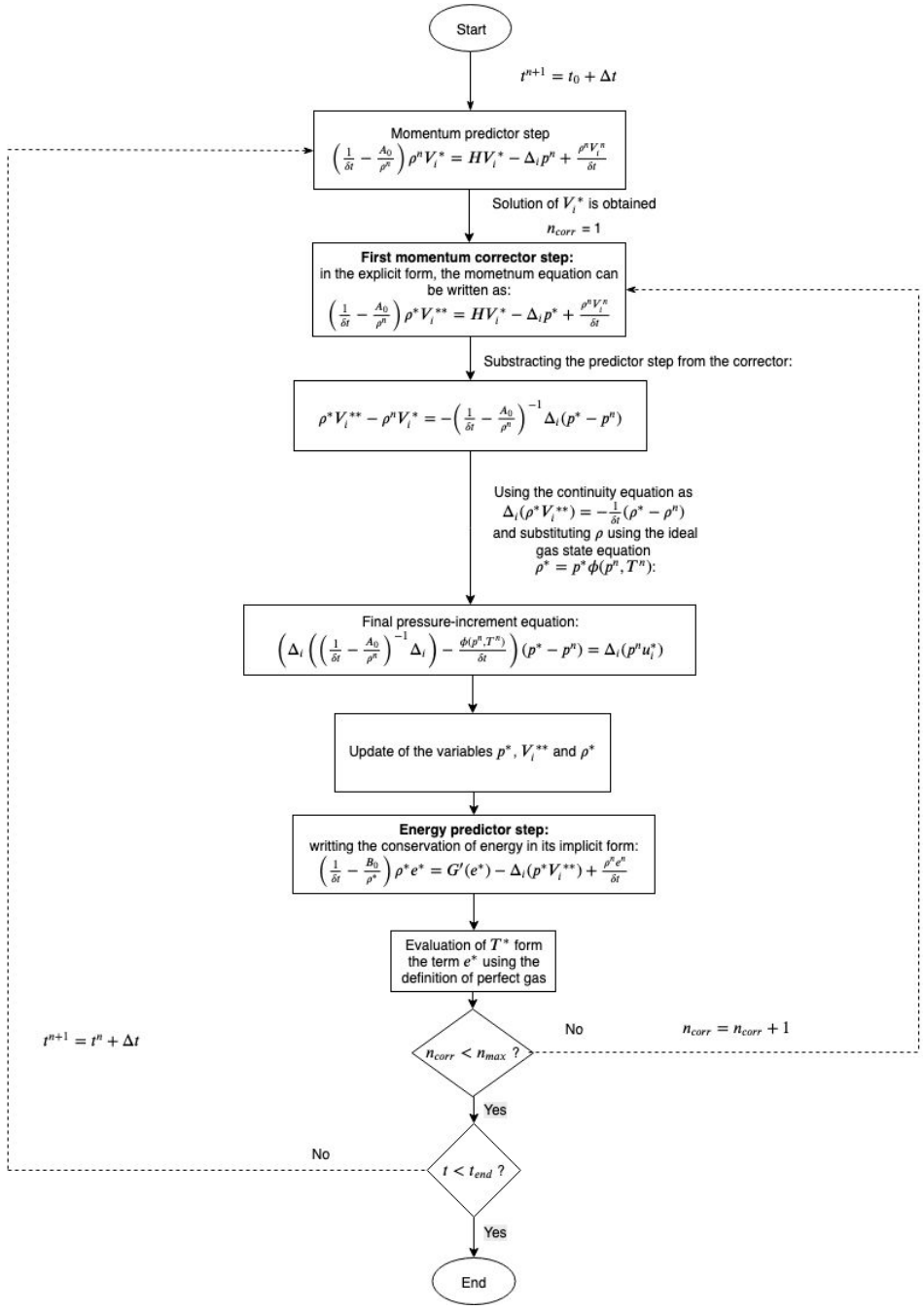


Figure 3.3: Algorithm of the *sonicFoam* scheme

3.3 Validation Case

In this section, a geometry similar to the decelerator will be analysed in order to setup the mesh and the solver conditions for the analysis of the aeroshell. The selected case will be extracted from (Trimmer, 1968). The problem is solved using the axisymmetric assumption, because is the one closest to the real wind tunnel conditions.

The reference paper mentioned above presents 4 different geometries to be analysed, each one undergone 3 trials at different Mach numbers. The probe selected for this validation study is the model A shown in Figure 3.4. Its dimensions are shown in Figure 3.5.

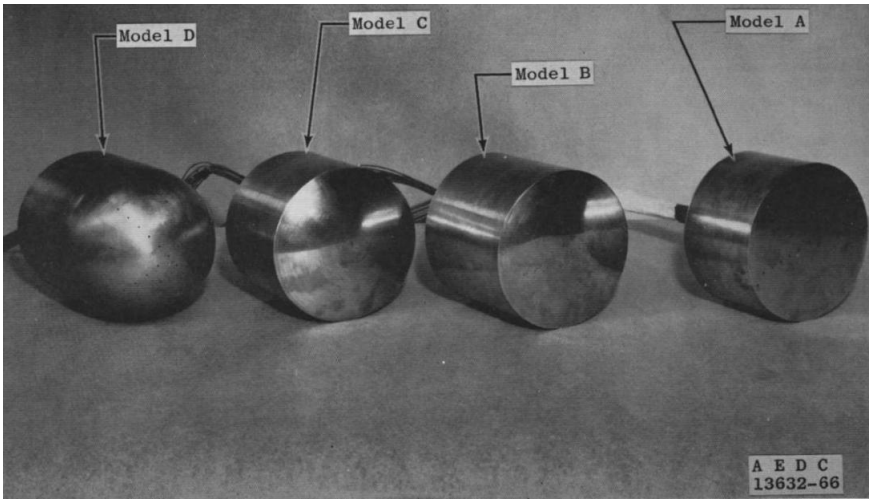


Figure 3.4: Different probe models. Source: (Trimmer, 1968)

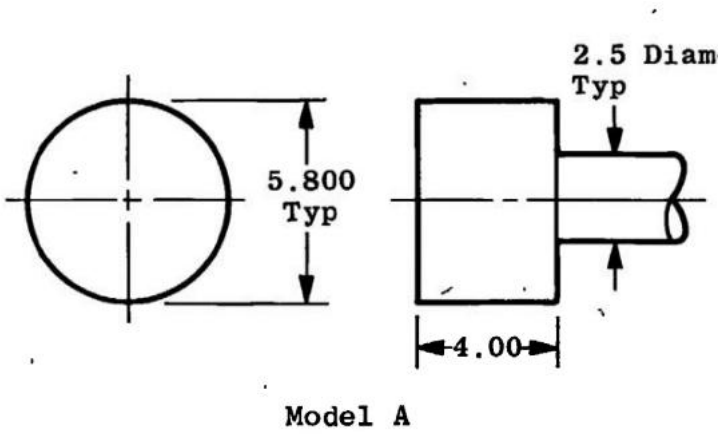


Figure 3.5: Dimensions of model A (in inches). Source: (Trimmer, 1968)

The main reason for selecting this geometry is because the forebody of the former and the aeroshell's are very similar.

The experimental data that will compared to the CFD results is the pressure measured along the forebody surface. In (Trimmer, 1968), this pressure has been measured with 19 differential transducers of 0.067 inches diameter each one. Although the measures presented several issues (low pressure gradients had to be measured at high pressure levels), the data has been measured several times, ensuring a very low dispersion and, therefore, making it reliable for a validation study.

The flow is a perfect gas that follows an isentropic expansion at the stagnation point: this assumption is shown valid in (Crawford, 1956) over an hemispheric geometry at M = 6.8 (after the shock wave, the fluid is subsonic and can be considered isentropic). The specific heats ratio γ_{heat} is 1.4

Some assumptions have been made in the present study:

- **Axisymmetric domain**: as the analysed probes have a symmetry over its longitudinal axis, it is possible to reduce the domain to a two dimensional case, neglecting lateral forces which will be considered equal. The governing equations used will be in the classical axissymmetric form.
- **Inviscid flow**: dissipative phenomena of viscosity, mass diffusion and thermal conductivity are neglected Anderson (1995). In this case, as the main aerodynamic parameters are produced by pressure gradients, this assumption is reasonable and allows the obtention of satisfactory results.

3.3.1 OpenFOAM solution scheme

The scheme selected for the solution of this problem is the **RhoCentralFoam** scheme. As shown in (Bondarev, 2018), it is the solver that achieves minimal error in each case. Although this result is based on the solution of a cone at zero angle of attack and at a lower Mach number, there are numerous coincidences of the conditions used (among them, the axisymmetric geometry and the supersonic inviscid flow), that indicates that this result can be applied to the present simulation case. More information about the most popular finite volume discretization methods and the internal structure of the solver can be found in Appendix A.

3.3.2 Compressible flow boundary conditions

According to (Greenshields et al., 2009), in a viscous Navier-Stokes simulation, the boundary conditions are not exact and can be difficult to completely determine. However, in an Euler approach, this conditions can be well-computed (Poinsot and Lele, 1992). In order to do so, it is important to take care of the wave-like behaviour of

the flow. In this sense, two cases have to be taken into account: if the wave enters to the domain, their dependence its outside the domain and therefore its information must be specified in the boundary. On the contrary, if the wave leaves the domain, its information its inside it and the value must not be specified.

Following this idea, at the **inlet** the values of U, p and T have been specified as they enter the domain. At the **outlet**, however, there is an upcoming pressure wave. Therefore, in order to deal with it, for the pressure p and the velocity V a non-reflective boundary condition is specified, with its freestream value imposed at least 30 diameters away (see (Versteeg and W, 2007)).

3.3.3 Description of the case

From the 3 analysis that have been carried on (Trimmer, 1968), it has been selected that at M = 10.12. The tunnel chamber conditions are shown below (Table 3.1), as well as the Reynolds number and the experimental stagnation pressure of the model p_0:

Model	M	Re	$P_s\ (Pa)$	$T_s\ (K)$	$P_0\ (Pa)$
A	10.12	$0.8 \cdot 10^6$	$8.27 \cdot 10^6$	1025	$2.34 \cdot 10^4$

Table 3.1: Tunnel chamber conditions (*source: L. Trimmer*)

This stagnation conditions have been converted into free-stream conditions using the compressible flow formulas shown in (Ames, 1951) (in this case, only equations 53 and 54 have been necessary, alongside with the ideal gas equation of state and the relation $M = \frac{V}{\sqrt{\gamma_{heat} RT}}$):

$$\frac{T_0}{T_\infty} = \frac{\gamma_{heat} - 1}{2} + M^2 + 1 \tag{53}$$

$$\frac{P_0}{P_\infty} = (\frac{\gamma_{heat} - 1}{2} + M^2 + 1)^{\frac{\gamma_{heat}}{\gamma_{heat}-1}} \tag{54}$$

yielding the free-stream boundary conditions of the case, which are $P = 180.04\ Pa$, $V = 1401.2\ \frac{m}{s}$ and $T = 47.71\ K$.

3.3.4 OpenFOAM structure

In order to solve this test case, a general structure has been followed which is common in most of OpenFOAM cases. Three folders are needed: *0*, *constant* and *system*, each of them containing the files shown in Figure 3.6.

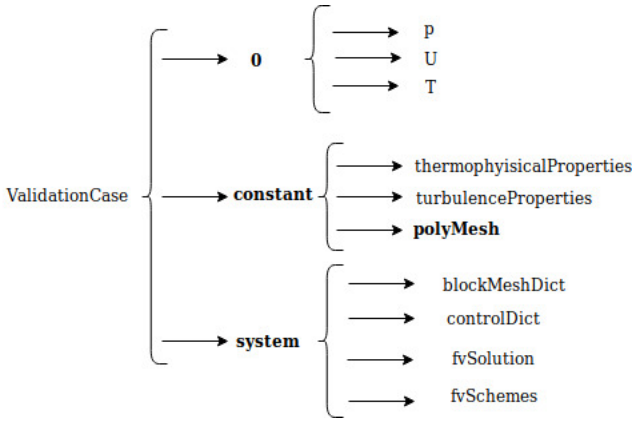


Figure 3.6: Structure of a general OpenFOAM case (folders are in bold).

These folders have the following function:

- **0**: the main physical parameters are defined, in this case, the velocity U, the pressure p and the temperature T.

- **constant**: gas properties are defined, such as turbulence model and thermophysical properties. Inside the *constant* folder, there is also stored the *polyMesh* folder, where all the data about the mesh is stored.

- **system**: here is where all the necessary files to control the case are stored. In this specific case, there is the *controlDict* file, which control the nummerical parameters, the *blockMeshDict* file, which sets the parameters for the creation of the boundary and the *fvSchemes* and *fvSolution* files, which specify the discretization schemes and the algorithm to solve each variable.

Each of these files are attached in the Appendix G.1.

3.3.4.1 Computational domain

The computational domain will be axisymmetric, as it accounts for the 3D effects of fluid flow. As the software used is OpenFOAM, the general form of the domain is the one depicted in Figure 3.7. This wedge has to straddle one coordinate plane, and it can't have an angle larger than 5° (this is because of computational precision).

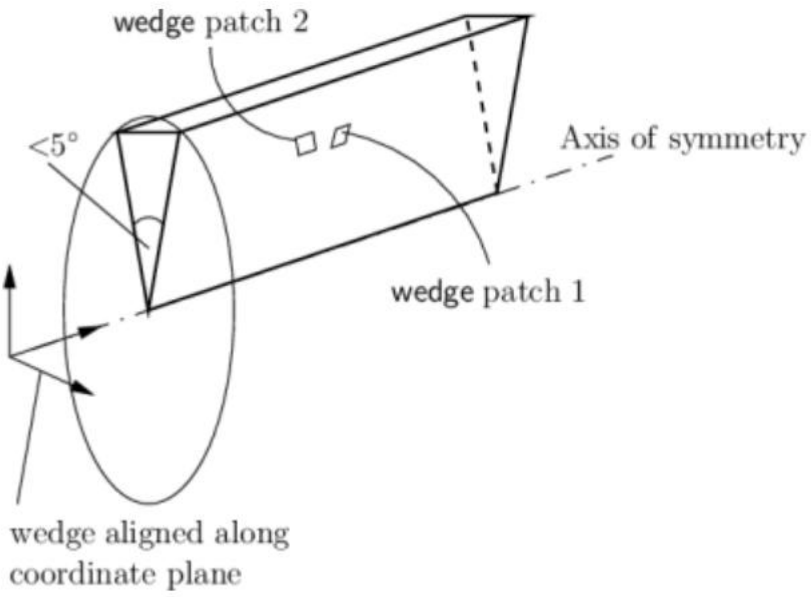


Figure 3.7: General setup of an axisymmetric doamin. Source: *OpenFOAM website*

Moreover, the *wedge* boundary condition is an special one that accounts for the rotational flow effect of an axisymmtric problem (i.e, it includes the axisymmetric term in the continuity equations, $\frac{\rho V_y}{y}$). Following this criterion, the domain adopted is shown in Figure 3.8, where the name of each boundary is specified.

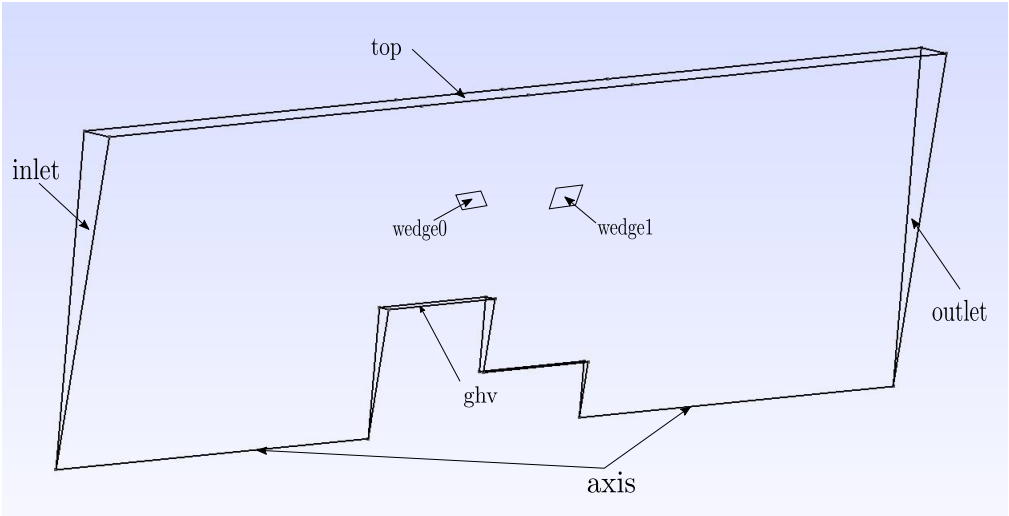


Figure 3.8: Computational domain

3.3.4.2 Mesh generation

In the initial stages of this work, several types of unstructured grids developed with the open source program *GMSH* have been tested with unsatisfactory results, due

to the poor aspect ratio and the high skewness. Therefore, the final mesh has been decided to be structured (as multiple papers present to be more succesful, such as (Bondarev, 2018), (Marcantoni et al., 2012) and others). This mesh has been done with the *blockMesh* utility of OpenFOAM, a library that creates hexahedral structured meshes for 2D and 3D problems. Figure 3.9 shows schematics of the domain:

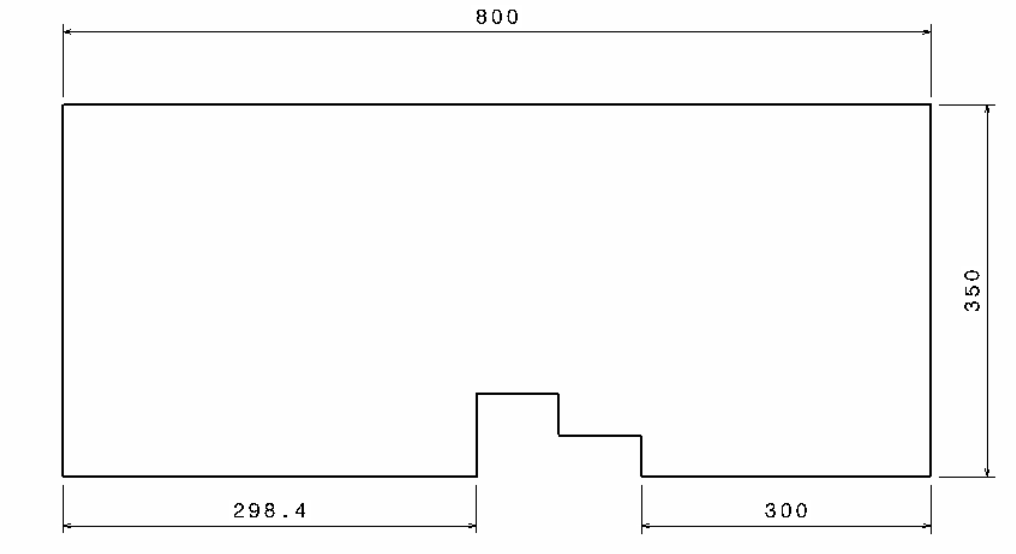


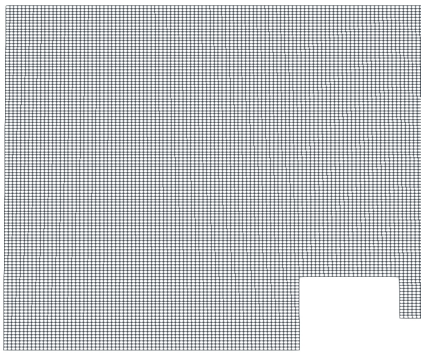
Figure 3.9: Computational domain, dimensions in mm. (The dimensions of the body are the same shown in Figure 3.5)

In this case, four meshes have been created. They have, from coarsest to finest, 22000, 88000, 135000 and 75000 elements. The last one has been graded towards the body surface. This feature requires the domain to be divided in blocks, therefore in order to increase the mesh density properly, the following criterion has been applied:

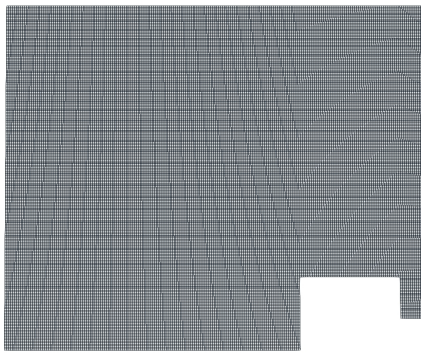
- Each block has been assigned a percentage of the total volume, depending on its size.
- The largest block has been assigned a suitable number of elements in its larger direction.
- As there are several interconnected block, it has been imposed that a direction shared by two blocks must have the number of elements.
- The resulting number of elements in each direction have been rounded to an even number for refining purposes.
- Between different meshes, the refining has been done with a scale factor, which ensured a proper increase of mesh density.

In the case of the graded mesh, after setting the desired number of elements, a grading has been performed towards the front and upper surface of the body, which are the most critical. As they appeared different sizes of cells in the boundaries of adjacent blocks, a local scale factor have been applied to these zones.

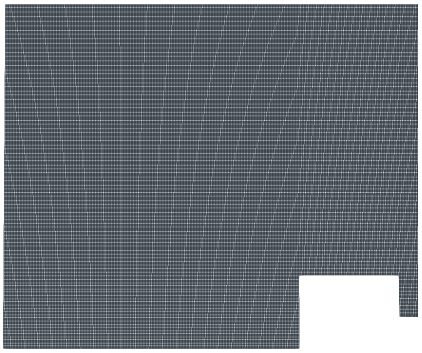
A detail of the mesh near the body surface is presented in Figure 3.10. The whole graded mesh is presented in Figure 3.11:



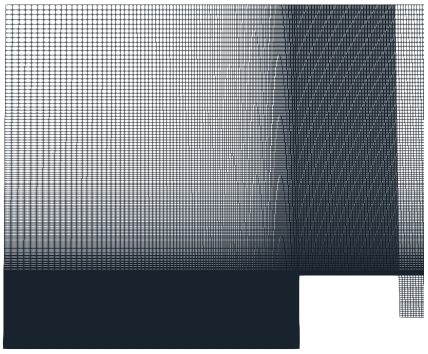
(a) Detail of coarse mesh



(b) Detail of fine mesh



(c) Detail of finest mesh



(d) Detail of graded mesh

Figure 3.10: Detail of each mesh used

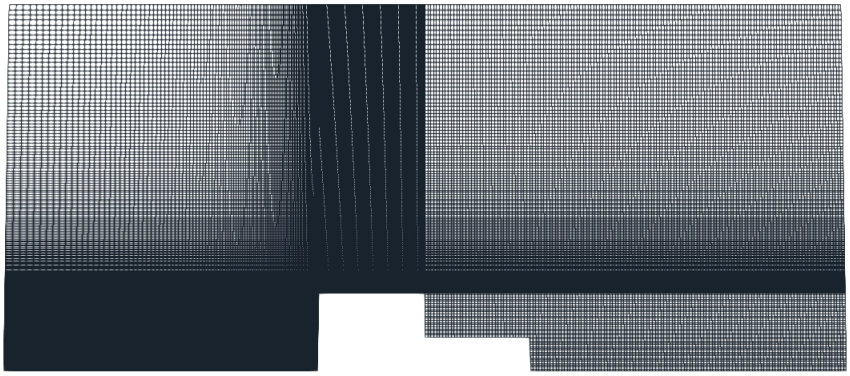


Figure 3.11: Graded mesh

3.3.4.3 Boundary conditions

The boundary conditions are specified in Figure 3.12. Each of them has the following meaning (for further information, see (Consulting, 2017)).

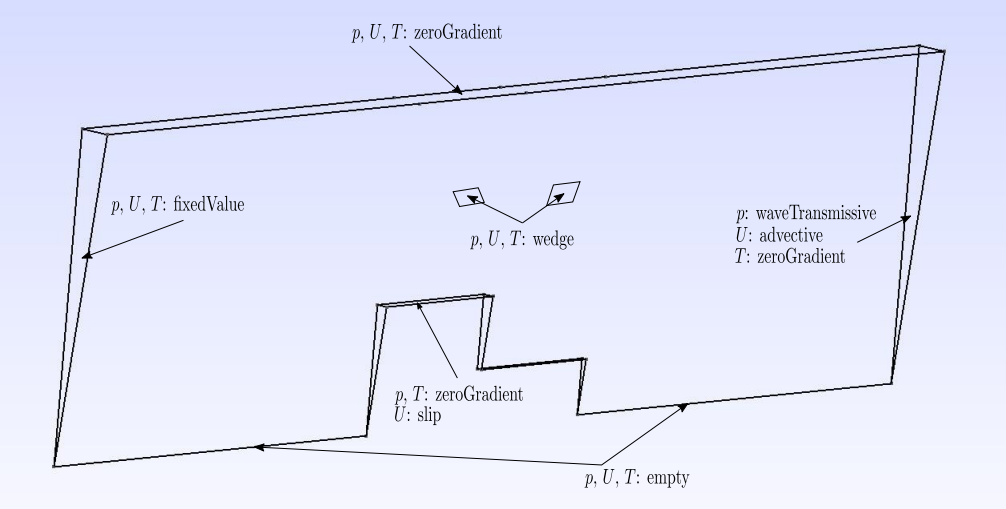


Figure 3.12: Assigned value for each boundary of the domain

Inlet

The upcoming flow is supersonic, therefore U, p and T must be imposed with *fixedValue* boundary condition, which specifies a constant value in the boundary.

Outlet

In this case, as there are shock waves inside the domain, it must be assured that these discontinuites are not reflected inside the domain. For doing so, a *waveTransmissive* boundary condition is imposed for the pressure. The key parameter is *lInf*, which specifies at which distance the pressure value has to be. According to (Versteeg and W, 2007), at a distance 30 times the obstacle characteristic length the flow is fully developed. In this case, as the radius $R = 0.074m$, the lInf has been set to $1.91m$. For the velocity, a similar boundary condition is imposed, named *advective* in OpenFOAM and which solves the equation $\frac{D\phi}{Dt} = 0$ at the boundary (see (Consulting, 2017)). For the temperature, a *zeroGradient* condition is imposed.

Top

As there is no variation of the fluid properties accross this boundary, a *zeroGradient* condition has been imposed for all the physical properties.

Wedges

In this case, the special *wedge* boundary condition has been imposed for all the fluid properties in order to allow the axisymmetric flux.

Axis

As this is the symmetry axis, it is only a delimiter of the domain. Therefore, the correct treatment in OpenFOAM must be the *empty* boundary condition, the same used in the delimiters of a 2D domain

ghv

Over the object, the pressure and the temperature do not change, therefore the *zeroGradient* condition is used. In the velocity, however, as this is an Eulerian approach of the Navier-Stokes equations, the *slip* boundary condition is imposed.

3.3.4.4 Schemes and solver controls

Before running the case, the solution schemes and control parameters must be defined. After several tests, the run time of the simulation have been established to be $0.0015s$, where it has been observed that the solution reached its stationary state (*rhoCentralFoam* is a transient solver and therefore the stationary state has to be defined with this

parameter). The time step is adaptive, as the important parameter that has been fixed is the Courant number (with this criteria, the simulated particles per unit time is the same in each cell):

$$Co = \frac{V\Delta t}{\Delta x} \tag{55}$$

where δx is the cell length. A $Co \leq 1$ assures that between two consecutive time steps, the particles will not change from one cell to another, which in the other hand will make the case to diverge. In order to obtain sufficient accurate results, a $C0 = 0.5$ has been set.

Regarding the resolution of the variables, it has been set that each of the different density variables are solved using a diagonal solver, which is a direct one and therefore the residual obtained will be 0. Two more variables have been specified, the velocity U and the total energy h. This is important in order to ensure a level of accuracy in this variables. Both are solved with a Gauss Seidel algorithm, which is the most reliable according to (OpenFOAM) and also supported by previous thesis such as (Bondarev, 2018). The tolerance of these variables has been incremented since a level of accuracy have been achieved, with a final value of 10^{-10}.

The selected schemes for each term of the Euler equations is the following:

- **Temporal schemes**: An *Euler* discretization is used, which is first-order accurate.
- **Gradient schemes**: For the gradient term ∇p, the *linear-upwind* scheme is used, which is second order accurate. The default option is *linear*, which is second order, but in combination with the upwind part, it can identify the direction of the flux and therefore has proven to give better results.
- **Divergence schemes**: For all these terms a *vanLeer* scheme is used, which is second order accurate and according to (Greenshields et al., 2009) give the most accurate results in high velocity compressible flows. Regarding the fields related with the velocity, OpenFOAM includes a version of the vanLeer with a "V". It smoothes the flow around corners in order to avoid spurious oscilations.
- **Interpolation schemes**: By default, the option is *linear*. However, in *rhoCentralFoam* it is necessary to reconstruct the variables (in compressible solvers, after solving the variables using a low-order scheme, the solution is reconstructed to achieve a higher order of accuracy). In this case, several schemes have been tested without achieving convergence. From all these classical schemes, such as *vanLeer*, *vanAlbada*, *Superbee*, *minmod*, the simulation diverged after a few iterations. The only one that was able to achieve convergende was the first order accurate *upwind* scheme.

Regarding the *fvSolution* file, the following schemes have been used:

- **Density and momentum and energy fluxes**: For these variables a classical diagonal solver has been selected.
- **Velocity and total energy**: For these two variables, the well-known Gauss-Seidel algorithm is choosen.

3.3.4.5 Numerical Results

All the simulations have been run until reach the stationary state, in this case, $0.0015s$. This value have been obtained experimentally: first of all, a large time step have been set and therefore, it has been adjusted to the mentioned value, as no oscillations in the fluid properties have been observed after it.

In order to compare the results obtained with the experiment, two graphs are presented: Figure 3.13 will compare the experimental results with the numerical ones exactly at the point of measurement (here, S is the distance from the axis to the measured point and S^* is the sonic point, which is supposed to be at the forebody edge). Figure 3.14 will compare the evolution of the adimensional pressure with the experimental one.

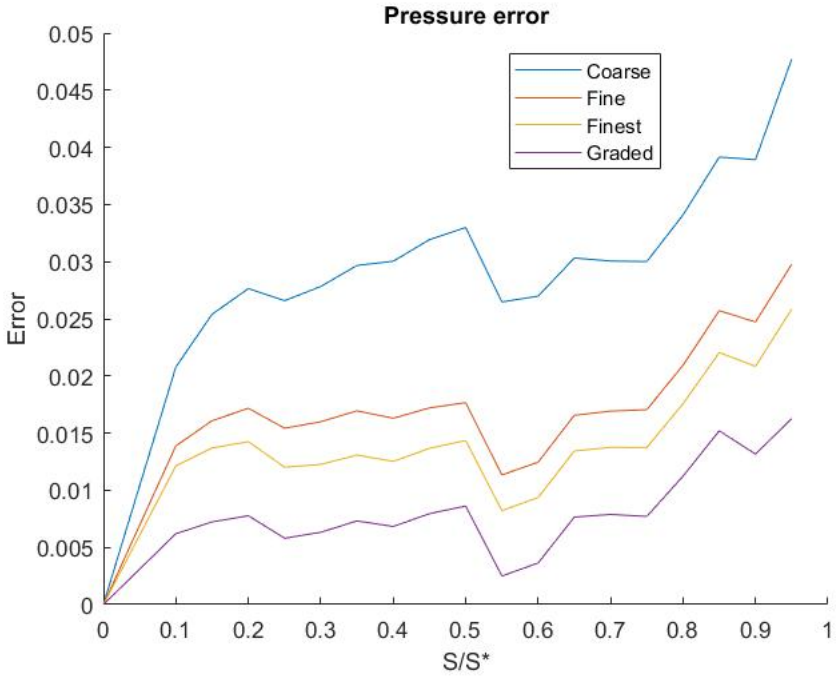
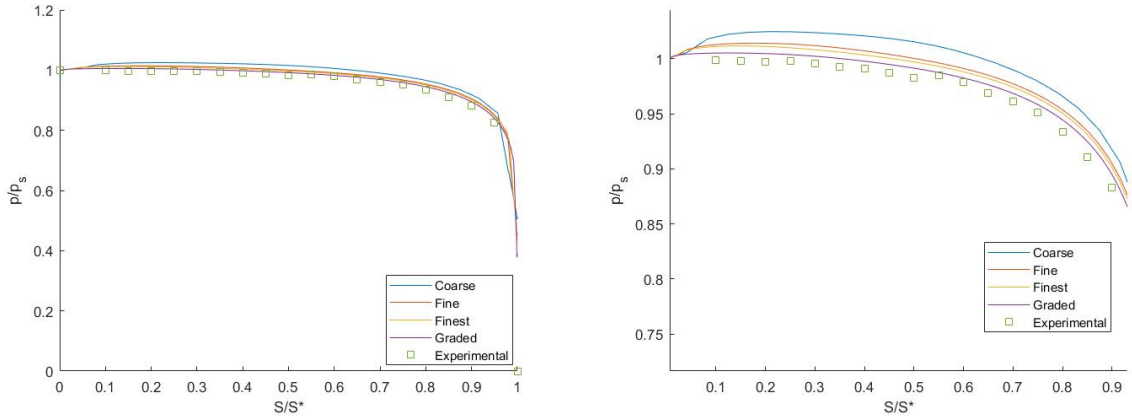


Figure 3.13: Relative error of each mesh



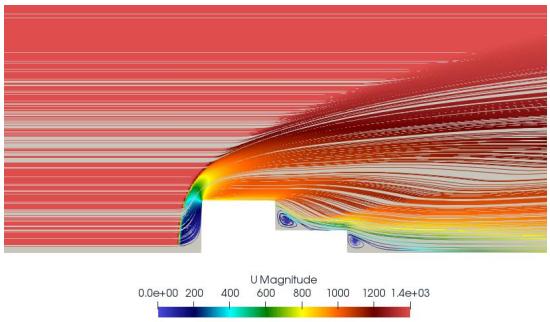
(a) Pressure distribution of each mesh

(b) Detail of pressure distributions

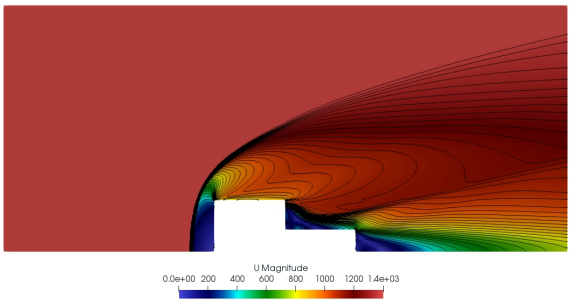
Figure 3.14: Pressure distribution of each mesh vs. the adimensionalised distance

Regarding the results, the best option is the graded mesh, because the maximum relative error is below 1 %. Moreover, its pressure distribution in the analysed face is the closest one to the experimental values, which demonstrates the mesh independency of the results (see Figure 3.14b, where it can be observed the convergence towards the experimental results). It can be seen that coarser meshes have pressure values higher than the stagnation pressure, and that this error is corrected when the mesh is refined and graded on the body surface.

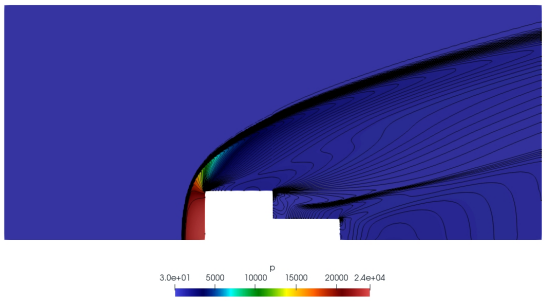
The results of the graded mesh are presented in Figure 3.15b for the velocity, Figure 3.15c for the pressure and Figure 3.15d for the temperature. The resulting streamlines are also shown in Figure 3.15a.:



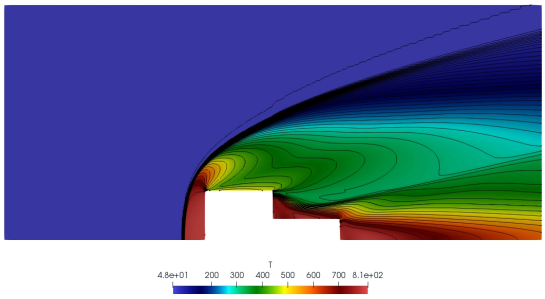
(a) Streamlines



(b) Velocity isolines



(c) Pressure isolines



(d) Temperature isolines

Figure 3.15: Results of the computational analysis of the validation case

Analysing the results in the pressure and the velocity, it can be observed how the front shock wave is formed. Moving backwards, at the end of the upper part of the obstacle, a set of expansion waves are formed due to the increase of section the fluid has to move forward. Due to this effect, and because it is an inviscid flow, a difference of velocites is produced in that point, causing an "inviscid wake" with vorticity (see Figure 3.15a): besides the Euler equations cannot present vorticity due to viscous effects, the fluid is not necesarily irrotational, and behaviours like the observed above can be produced if these separation condition happens (see Appendix C for more information about this behaviour). Finally, while the wake is being formed, the flow accelerates after the bow shock and a second shock wave is formed at the end of the domain, outside the wake. This wave is weaker than the previous one.

Regarding the temperatures, a maximimum of 800 K is observed in the front face of the model. Its effect on the probe behaviour is commented in the aeroshell analysis.

3.4 Aeroshell analysis

After the validation case, the aeroshell analysis is carried out. Two different geometries with different cone angles (70° and 45° deg) are studied with the objective to calculate the aerodynamic parameters required for the flight simulator. For each geometry, two analysis will be done: the first will assume a ballistic descent, and, thus, axisymmetric flow. The second will consider a reentry at $AoA \neq 0$. In this case, for the sake of simplicity, the flow is considered 2D.

3.4.1 Description of the case

The objective of this analysis is to extract all the necessary coefficients to generate the interpolation curves that are going to be included in the flight simulator. In order to do so, a test matrix has been created. The design points have been obtained from the paper (Pasolini et al., 2017), shown in Figure 3.16. The data of temperature and density have been extracted from the atmospheric model presented in the Chapter 2.

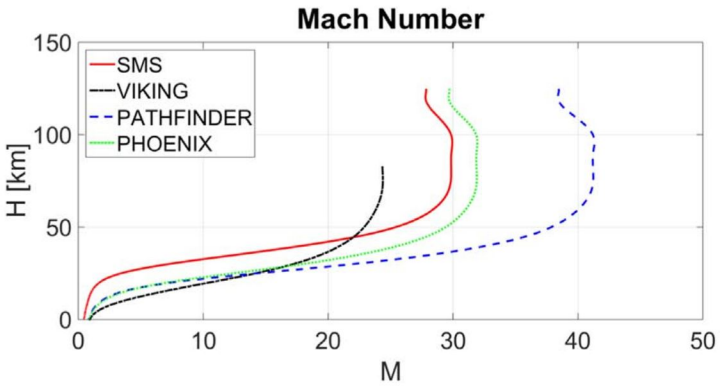


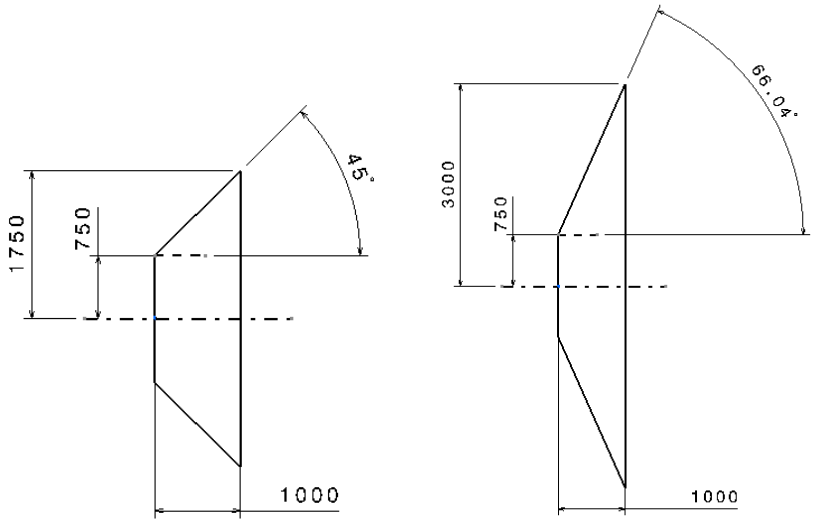
Figure 3.16: Mach number vs. Velocity (source: (Pasolini et al., 2017))

With this data, and using the ideal gas equation of state and the definition of Mach number, the test matrix can be created (Table 3.2):

M	$H\ (m)$	$T\ (K)$	$\rho\ (\frac{kg}{m^3})$	$P\ (Pa)$	$V\ (\frac{m}{s})$
0.8	10502.5	184.59	0.0058	205.51	172.29
0.95	13350.8	192.13	0.0043	156.71	208.73
1.5	18735.7	192.75	0.0026	95.3	330.1
1.9	20666.2	189.35	0.0021	76.26	414.43
2.4	22539.2	186.26	0.0018	62.57	519.19
3.5	25096	183.076	0.0014	47.5	750.65
5	27533.1	181.91	0.0001	35.85	1068.92
7.5	30562.9	180.55	0.00078	27.02	1507.4
10	32132.7	176.79	0.00066	22.54	2107.6
12	34324.5	172.67	0.00052	17.36	2506
15	37028	172.67	0.00039	12.92	3124.4
20	41975.6	170.19	0.00023	7.41	4135.8
25	49095.3	190.8	0.000099	3.11	5065

Table 3.2: Trials matrix of the selected design points

As mentioned before, two different geometries will be analysed. The dimensions of each geometry are depicted in Figure 3.17.



(a) Aeroshell, D = 3.5 m (b) Aeroshell, D = 6 m

Figure 3.17: Body geometries

The analysis will be done for $\alpha = 0$ in the axisymmetric cases and for a range of $\alpha = [0, 2, 4, 8]$ for the 2D cases. The boundary conditions, the schemes and the analysis settings used are the same as in the validation case, where they proved to obtain good results.

Regarding the mesh, the number of elements and the procedure to create it has been maintained, but several adjustments have been introduced, specially due to the difference of the body and the multiple Mach numbers analyzed (i.e, the grading and local adjustments due to different-sized elements). This was very important because of the position of the shock wave.

3.4.2 Preliminary analysis of the flow field

Before running the case, a literature study has been done in order to identify the expected flow characteristics in a high speed flow simulation (see (Mehta, 2006), (Viviani et al., 2012) and (Prasad and Srinivas, 2014)).

For a generic blunt body at supersonic speeds, typical flows present the characteristics shown in Figure 3.18. Firstly, a bow shock appears in front of the body nose.. After this discontinuity, there is a subsonic zone, but due to the curvature of the shock the flow is capable of turning supersonic again (the sonic line indicates the isoline where the $M = 1$). These type of shocks move closer to the body as the Mach number increases (see Appendix B for information about these behaviour).

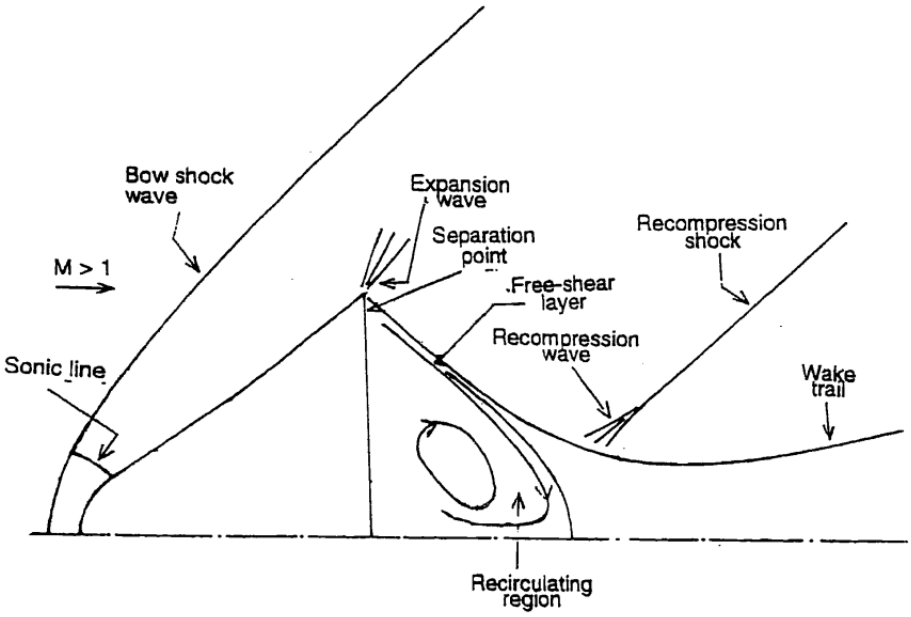


Figure 3.18: Flow around a blunt body (source: (Mehta, 2006))

When the flow arrives at the afterbody, an expansion wave is produced. Due to this expansion, a difference of tangential velocities is created and a high value of vorticity is generated, producing a recirculating inviscid flow. This is possible because the condition of irrotationality is not given in Euler equations. In the Appendix C, a famous test is shown which demonstrates this fact.

If the free-stream flow has a high Mach number (typically $M > 3$), a recompression shock is created. It has a much lower strength than the first one.

3.4.3 OpenFOAM solution structure

The structure of this case is almost the same that presented in Section 3.3. The different files that have been added to solve the configuration are underlined in red in Figure 3.19.

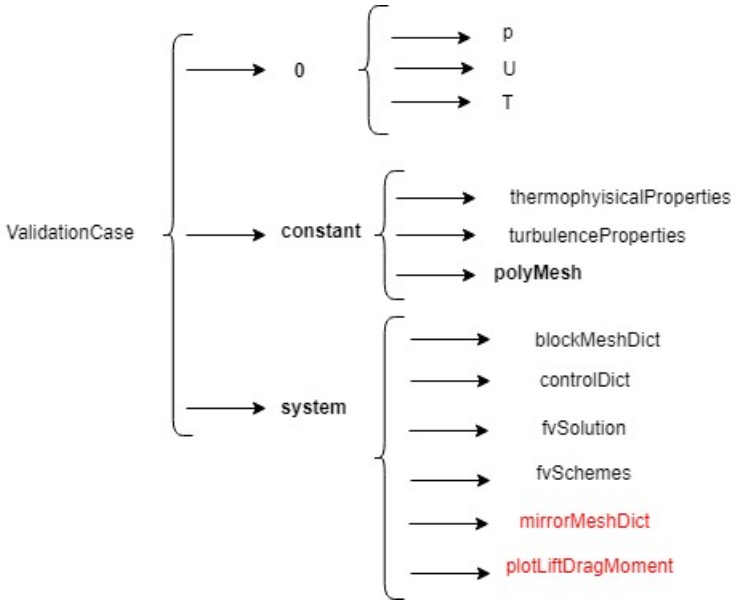
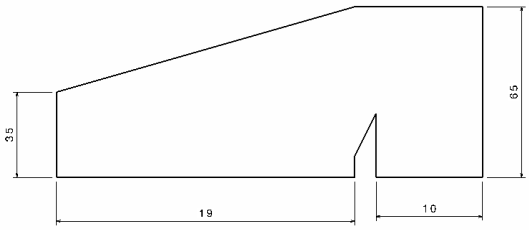


Figure 3.19: File structure inside the aeroshell case

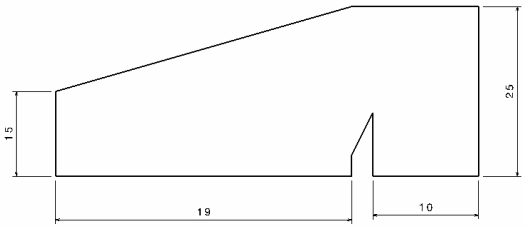
The new files added are *mirrorMeshDict* and *plotLiftDragMoment*. The former is an OpenFoam utility that allows to mirror meshes, and since the studied body is symmetrical, this utility is going to be used for the 2D analysis, where only half of the mesh needs to be created. The latter is a script that calls two libraries of OpenFOAM that calculate the aerodynamic forces and coefficients over a selected patch (see Casacuberta (2014)). The calculation of these coefficients will be explained in the results section. The files presented in Figure 3.3 can be found in Appendixes G.3.

3.4.4 Computaional domain and mesh parameters

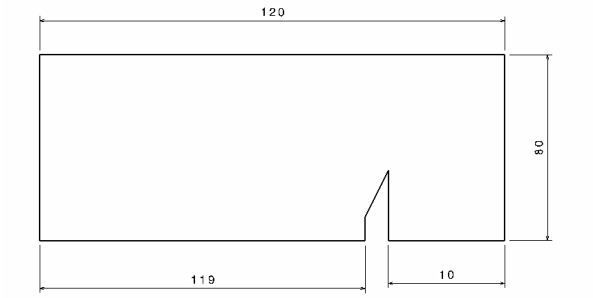
Two types of domain are used in this study: a wedge like the domain for the validation case and another rectangular, for the 2D analysis. As the analysis will be done in different flow regimes, a total of three different domain need to be defined: one will be used for the supersonic and hypersonic cases, and the other for the transonic analysis. This is necessary because in transonic flow, discontinuities are produced in every direction. They are presented in Figure 3.20:



(a) 2D hypersonic domain, in m



(b) Axisymmetric hypersonic domain, in m

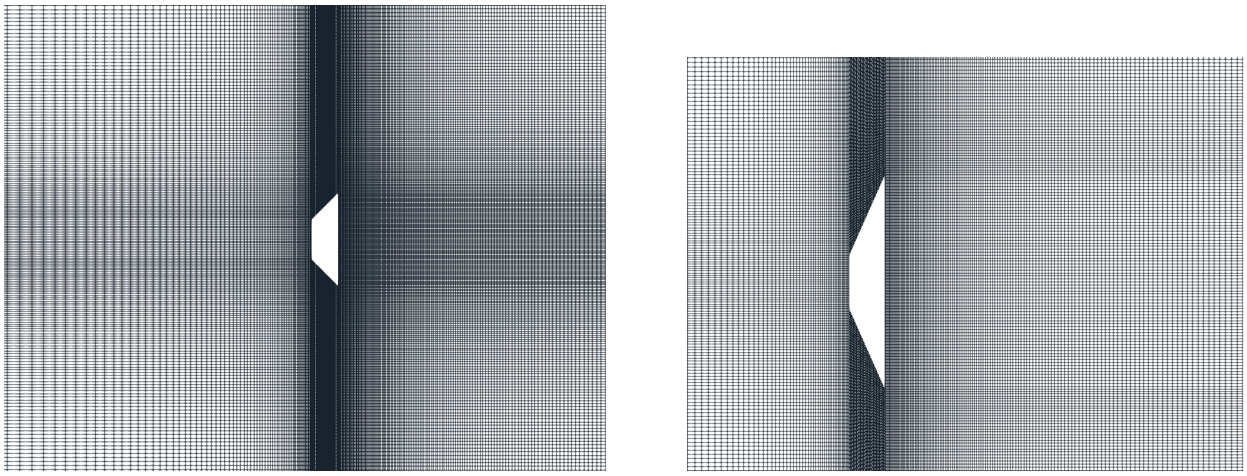


(c) Dimensions of the transonic domain, in m

Figure 3.20: Computational domains used in the analysis

Regarding the mesh, the procedure to generate it has been the same followed in the validation case, paying special attention in maintaining the number of elements. Nonetheless, two aspects should be noted: first of all, an special adjustment needed to be done in the cone's oblique face, because if not the elements had a high skewness. Secondly, in the 2D case it has been considered that, as the mesh was mirrored, the elements would be too. Therefore, for the 2D case there were 148392 elements and for the axisymmetric case, 74196 elements.

In Figure 3.21 there is a detail of the mesh used for each geometry, as the number of elements and grading were equivalent for all the presented domains:



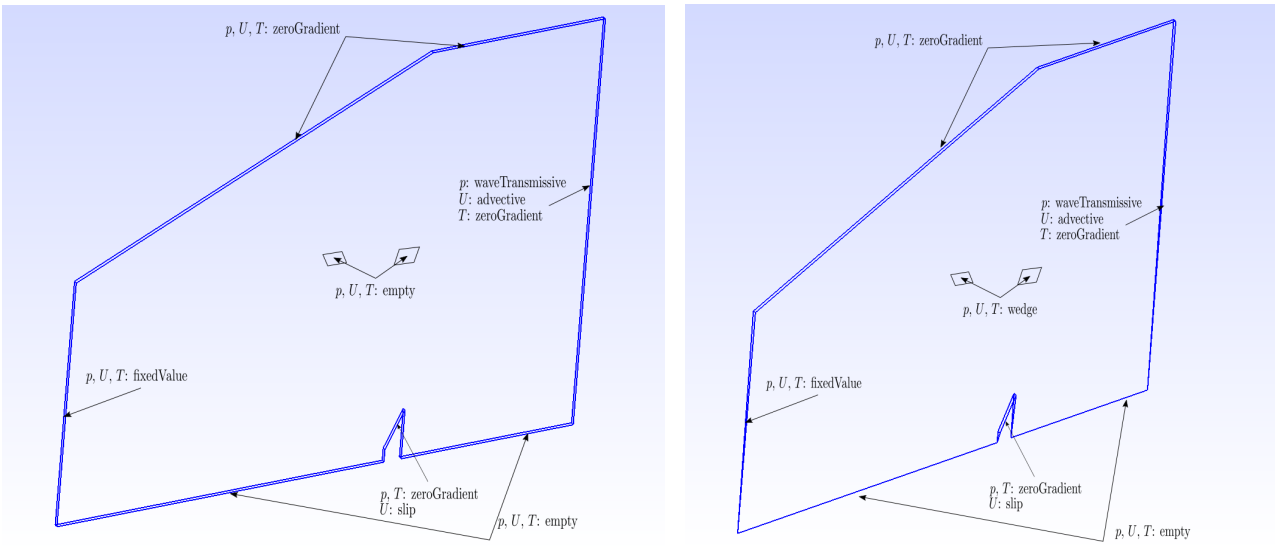
(a) Detail of the mest for the D = 3.5 m cone

(b) Detail of the mest for the D = 6 m cone

Figure 3.21: Mesh details for each geometry

3.4.5 Boundary Conditions

The boundary conditions are practically identical to those used in the validation case. The only remarkable differences are that, in the 2D case, the *wedge* boundaries need to be replaced for the *empty* ones; and that, in the reflexive boundaries (i.e, *advective* and *waveTransmisse*), the *lInf* parameter needed to be changed in order to adjust to each geometry diameter (170 metres for the $D = 6m$ aeroshell and 90 metres for the $D = 3m$ aeroshell). An schema of the boundaries for both the 2D case and the axisymmetric are presented in Figure 3.22:



(a) Boundaries of the 2D case

(b) Boundaries of the axisymmetric case

Figure 3.22: Boundary conditions

3.4.6 Schemes and simulation control

As well as in the domain and the meshing, several adjustments needed to be done in the schemes used.

The solver *rhoCentralFoam* has been used for all the supersonic and hypersonic cases, as it has been proven to be the fastest and most accurate solver for this type of flow. The files *fvSchemes* and *fvSolution* are exaclty the same used in the validation cases.

However, the two lower transonic cases (both M = 0.8 and M = 0.95) need to be solved in a different manner. As stated in (Morawetz, 2004), transonic flows involve regions of subsonic and supersonic flows, and the governing equations are elliptic and hyperbolic, respectively. Thus, the solver needs to be changed. Regarding (OpenFOAM) pressure based solvers are well suited for subsonic compressible flows. In this case, the *sonicFoam* solver has been used, which is a transonic-supersonic solver for compressible fluids based in the PISO scheme. The details of its behaviour are described in Appendix D.2.

First of all, the *fvSchemes* has been defined. The first iterations showed that with a discretization scheme in convective terms based on VanLeer limiters the simulations diverged during the first time steps. After several trials, it has been observed that the simulations diverged due to the unboundedness of the scheme. Therefore, two different kind of schemes have been used: *limitedLinear* and *Gauss upwind*. The first was sec-

ond order accurate, and it had and adjustabale bound, which prevented the results to diverge. The latter was first order accurate and very robust, which also prevented the divergence. After several iterations, the *Gauss upwind* proved to be the most stable and to give the best results. The rest of the schemes (divergence and interpolation) were kept at *linear* discretization.

On the other hand, the *fvSolution* file differed from the described in the validation case. In this case, as *sonicFoam* is an iterative pressure based solver, the nummerical methods used to solve the variables are different and new parameters need to be specified.
First of all, the variables to be solved are discretised as follows:

- **Pressure and density**: The solver used is a preconditioned bi-conjugate gradient (PBiCG) with a symmetric DIC, which accelerates the convergence.
- **Velocity and density**: Here, the same diagonal solver as the solutions file in *rhoCentralFoam* has been used

In all these variables a tolerance of 10^{-6} have been imposed, which ensures a good accuracy whilst the computational cost is not too high.

The general scheme (PISO or PIMPLE) have been defined. In several thesis about transonic CFD modelling (see (Nikaido, 2015)) it is claimed that the PISO scheme provides acceptable results. Thus, as a PIMPLE-based algorithm requires an outer loop which elevates the computational cost, a PISO three-stage scheme has been adopted. Finally, as the mesh showed some non-orthogonality, three non-orthogonal correction loops have been imposed (mainly, for a low value of non-orthogonality 1 or 2 correctors are needed, while for high-non-orthogonal meshes, a maximum of 20 provides good results).

Another important variable was the simulation time, defined in the *controlDict* file. The necessary time in order to reach the stationary state was determined experimentally. The values are shown in Table 3.3:

M	$t_{stationary}$ (s)
0.8	2.5
0.95	2.5
1.5	1.5
1.9	1
2.4	0.4
3.5	0.16
5	0.12
7.5	0.1
10	0.08
12	0.06
15	0.04
20	0.04
25	0.04

(a) Stationary time - 2D case

M	$t_{stationary}$ (s)
0.8	1.5
0.95	1.5
1.5	0.7
1.9	0.5
2.4	0.4
3.5	0.16
5	0.12
7.5	0.1
10	0.08
12	0.06
15	0.04
20	0.04
25	0.04

(b) Stationary time - axisymmtetric case

Table 3.3: Stationary times for each case

As it can be seen, all the times are practically equal except those below $M = 2$. This because it was observed that the axisymmetric case reached its stationary state before, and in order to save computaional time, the time of the simulation was reduced.

3.4.7 Computation of aerodynamic coefficients

OpenFOAM provides an utility to export aerodynamic coefficients while the simulation is being runned. For using it, the *controlDict* file needs to be edited and call the *plotLiftDragMoment* function (in this case, it will be extracted from (Casacuberta, 2014)). This function has two different parts: in the first one, it calculates the forces acting over the selected patch elements. The second one calcualtes, using freestream values of velocity and density, and with reference area, the lift and drag coefficients. In this project, as said in a previous section, it has been decided to maintain the domain and change the velocity components in order to simulate the AoA. Therefore, in each AoA case the direction of the lift and drag had to be computed properly. Finally, regarding the reference area, and depending on the domain:

- 2D domain: the reference area adopted is the body diameter times the width of the domain (0.2 m in all cases).
- Axisymmetric domain: the reference area is the corresponding to a circular section of the wedge total angle (in this case, 5°).

The moment coefficient has been calculated from scratch. The reason is that OpenFOAM showed discrepancies between the expected values and the calculated ones.

Therefore, after setting the position of the CG, the position of the CP has been calculated as

$$CP = \frac{\int p(x_{icell} - x_{inose}) \cdot dx_i}{\int p \cdot dx_i} \tag{56}$$

where x_i is, in this case, the X and Z coordinate of each surface cell with respect the CG. Finally, the moment coefficient can be calculated using Equation 57:

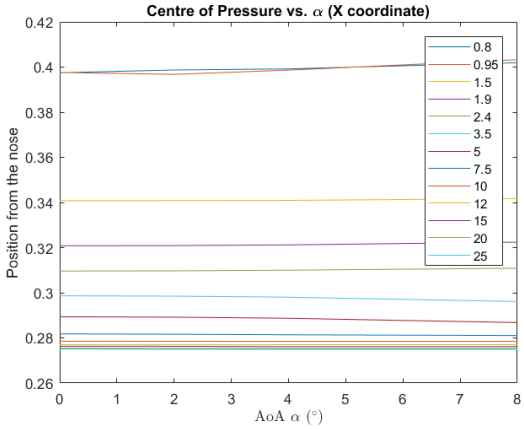
$$C_M = \frac{x_{CP} - x_{CG}}{D} (C_L \cos\alpha + C_D \sin\alpha) + \frac{z_{CP} - z_{CG}}{D} (C_L \sin\alpha + C_D \cos\alpha) \tag{57}$$

The structure and internal functions are shown in Appendix F.

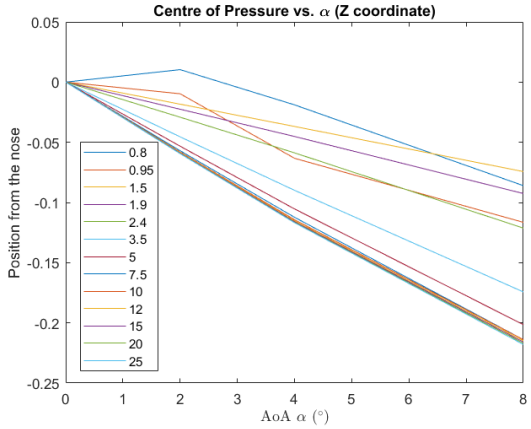
3.4.7.1 Position of the CG with respect the CP

In a reentry of a capsule configuration without elevons, the stability becomes of critical importance. In general, a stable flight occurs when $\frac{\delta C_m}{\delta \alpha} < 0$, because an increase of AoA produces a stabilizing moment to reduce it. The stable position (i.e, trim condition), occurs the $\frac{\delta C_m}{\delta \alpha} < 0$ and $C_{mC}G = 0$ (Schoenenberger and Queen, 2019). For a symmetrical body, this occurs at $\alpha = 0$.

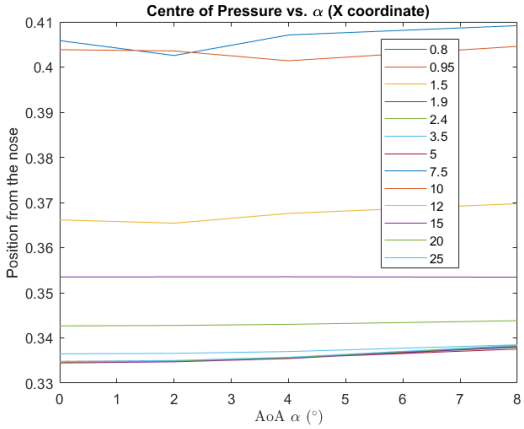
In order to obtain a $\frac{\delta C_m}{\delta \alpha} < 0$, the CP should be positioned **behind** the CG. After calculating the centre of pressure at all Mach numbers and at all AoA, these are the positions of the CP:



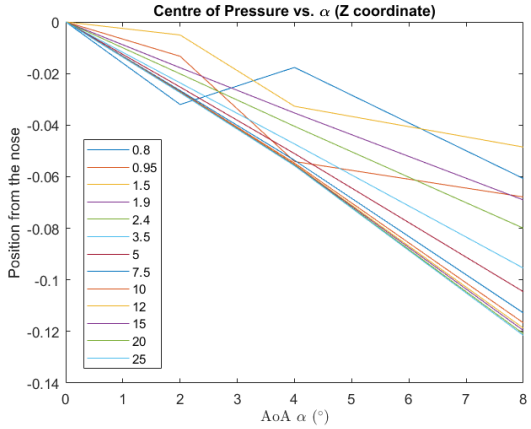
(a) CP position on the X direction, D = 3.5 m



(b) CP position on the Z direction, D = 3.5 m



(c) CP position on the X direction, D = 6 m



(d) CP position on the Z direction, D = 6 m

Figure 3.23: Positions of the centre of pressure for both geometries at different Mach numbers

As it can be seen in Figure 3.23 , the longitudinal position of the CP ranges from 0.33 to 0.41 metres approximately (measured from the forebody). As the CG must be advanced with respect the CP , it has been decided to put it at **0.25 metres**. Further analysis showed that at that position the capsule was stable and its pitching moment derivatives were acceptable. As the capsule is symmetric, $Y_{CG} = 0$.

3.4.8 Numerical Results

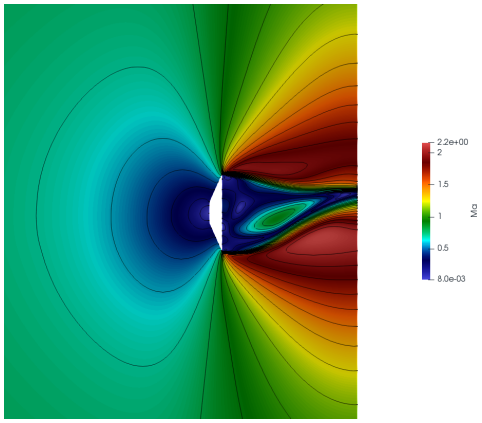
The results of the analysis carried out in the aerodynamic analysis are presented. For the sake of simplicity, only some analysis at certain Mach numbers and AoA are presented. For each geometry, the results displayed are temperature, pressure and Mach number. Three Mach numbers, one for each stage of the flow, have been selected: $M = 0.8$ for the transonic phase, $M = 2.4$ for the supersonic phase and $M = 7.5$ for

the hypersonic phase.

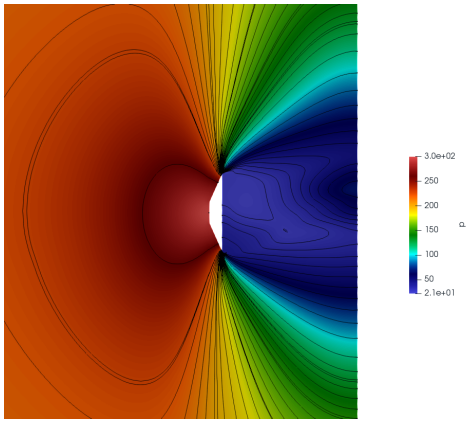
Finally, in order to study the effect of angle of attack, two angles of attack will be taken, $\alpha = 2°$ and $\alpha = 8°$, as well as the axisymmetric case, which is at $\alpha = 0°$.

3.4.8.1 2D case

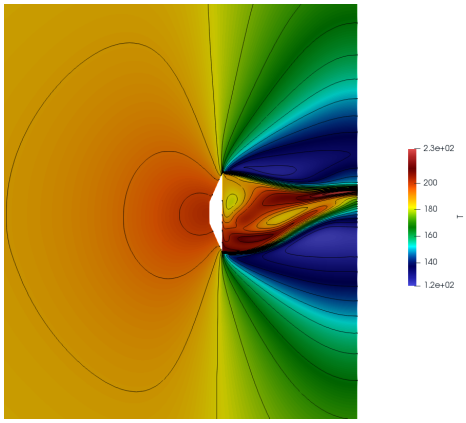
Regarding the $D = 6m$ capsule, the results at $\alpha = 2°$ and $\alpha = 8°$ are plotted in Figure 3.24 for the transonic phase:



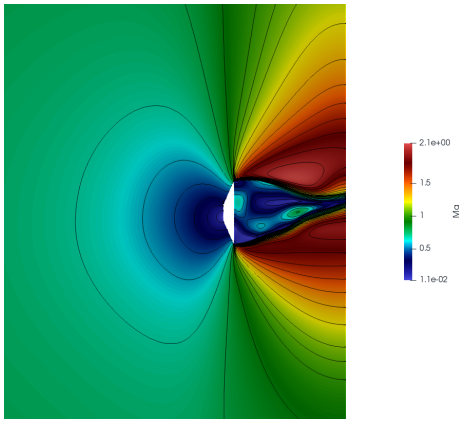
(a) Mach number, $\alpha = 2°$



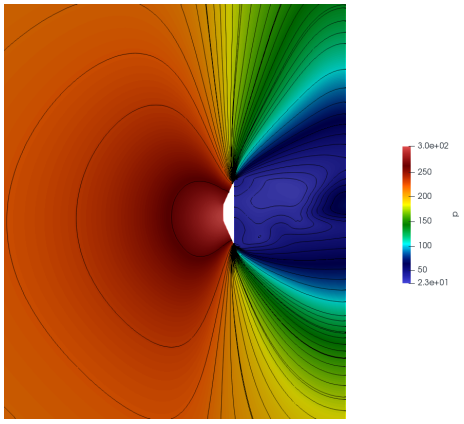
(b) Pressure, $\alpha = 2°$



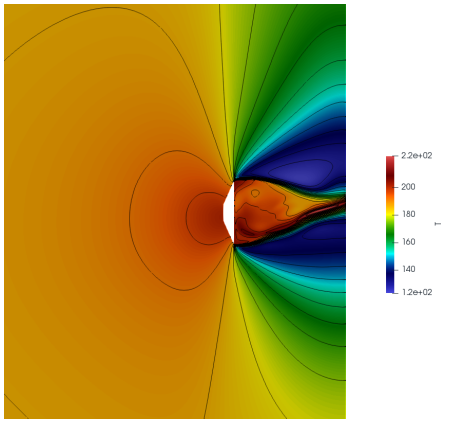
(c) Temperature, $\alpha = 2°$



(d) Mach number, $\alpha = 8°$



(e) Pressure, $\alpha = 8°$



(f) Temperature, $\alpha = 8°$

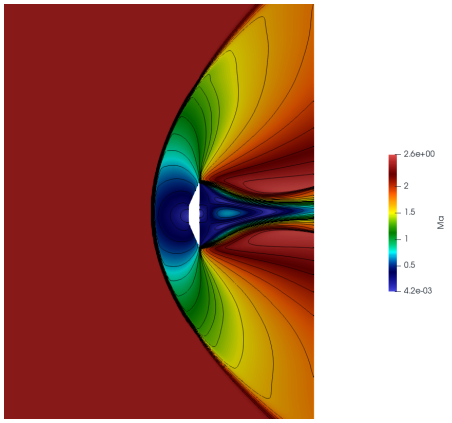
Figure 3.24: Results on temperature, pressure and Mach number for $M = 0.8$ and $D = 6m$

In Figure 3.24, it can be observed that the wake, while they can'tbe fully analysed because this is an inviscid approach, they show a periodic behaviour, which is predicted in several studies for transonic flow for the stationary state (see (Moretti et al., 1987)). Despite this oscillating behaviour, the effect in the aerodynamic forces was not important. Despite the flow is accelerated, no shock-wave is produced in the wake. And finally, comparing both angles of attack, it can be seen that as the AoA increases, the stagnation point moves downward.

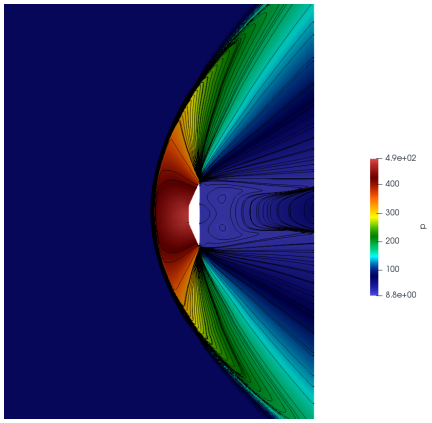
Regarding the pressure, the values obtained are reasonable (Moretti et al., 1987). The behaviour of the isonlines show an increase of the pressure towards the direction of the flow, which is expected (similar results have been obtained in (Nikaido, 2015)).

Analysing the Mach number, an increase up to $M = 2.2$ is observed. These values are obtained in the wake, and more iterations should be done in order to calculate them properly. Finally, the temperature plots do not show a high temperature gradient, which at low velocities is expected.

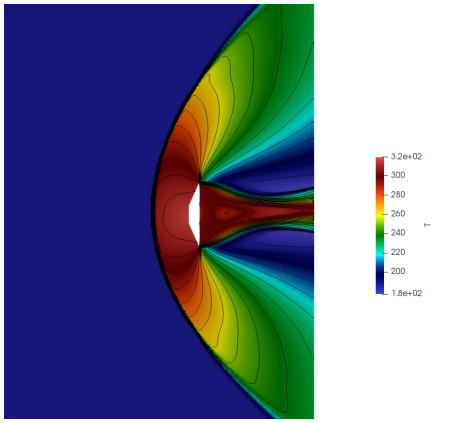
Following the transonic results, Figure 3.25 and Figure 3.26 show the results of the supersonic phase and the hypersonic phase, respectively:



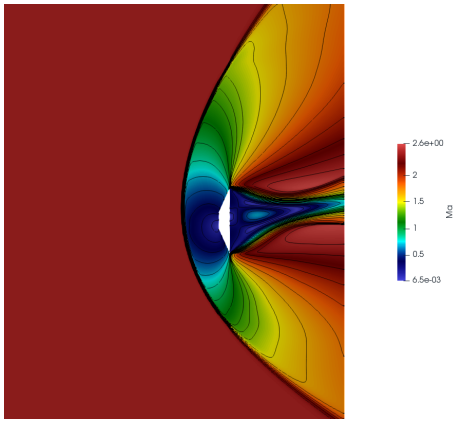
(a) Mach number, $\alpha = 2°$



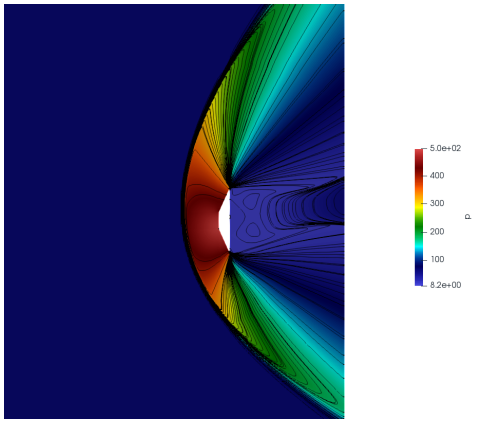
(b) Pressure, $\alpha = 2°$



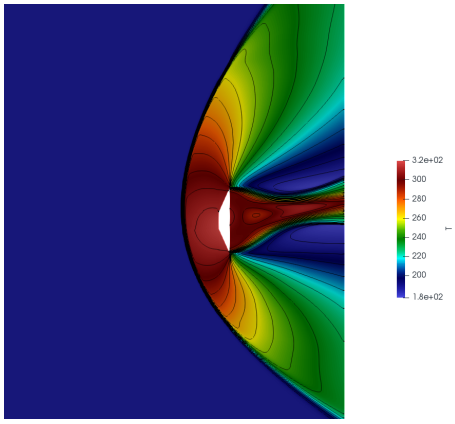
(c) Temperature, $\alpha = 2°$



(d) Mach number, $\alpha = 8°$

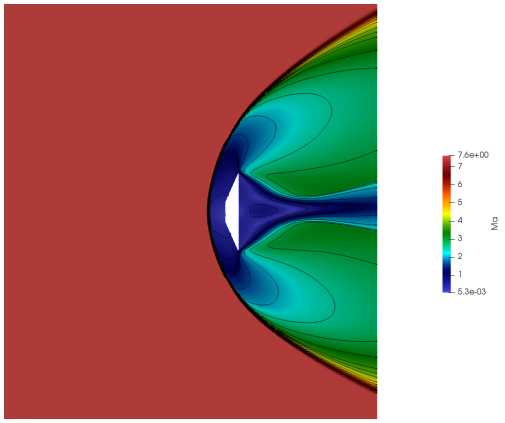


(e) Pressure, $\alpha = 8°$

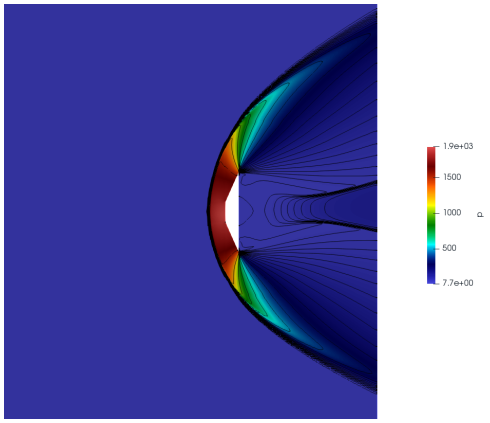


(f) Temperature, $\alpha = 8°$

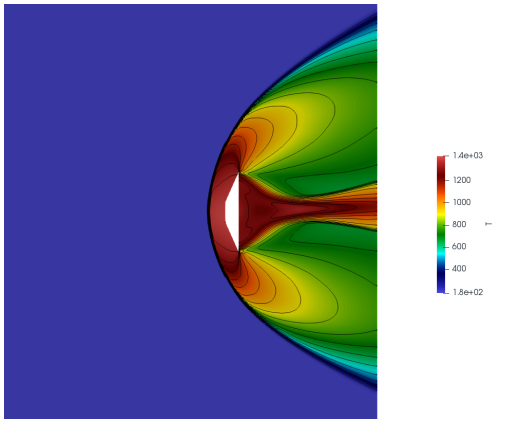
Figure 3.25: Results on temperature, pressure and Mach number for $M = 2.4$ and $D = 6m$



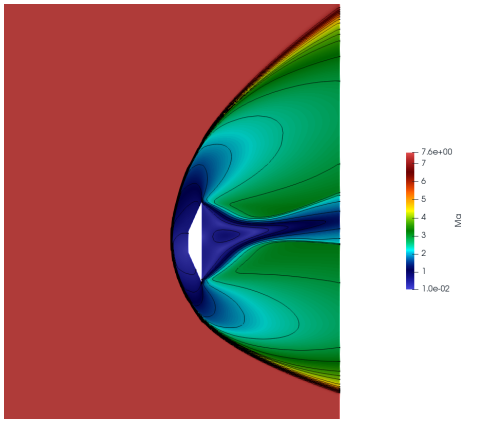
(a) Mach number, $\alpha = 2°$



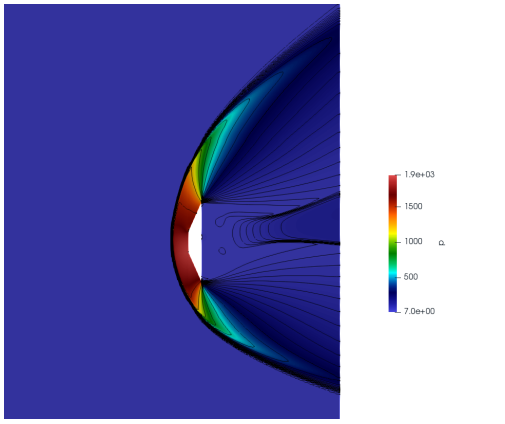
(b) Pressure, $\alpha = 2°$



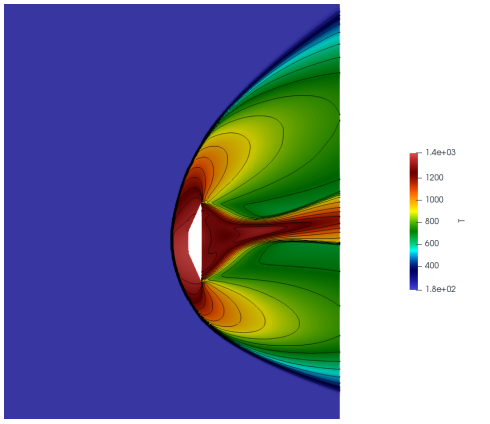
(c) Temperature, $\alpha = 2°$



(d) Mach number, $\alpha = 8°$



(e) Pressure, $\alpha = 8°$



(f) Temperature, $\alpha = 8°$

Figure 3.26: Results on temperature, pressure and Mach number for $M = 7.5$ and $D = 6m$

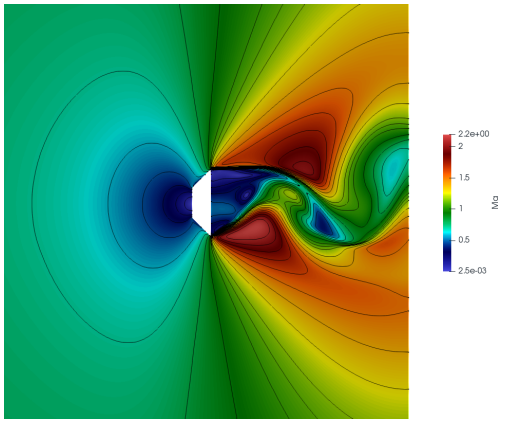
The main difference between the supersonic and the hypersonic results is the shock wave behaviour. First of all, as presented in (Sinclair et al., 2017), the standoff distance of the shock wave in the supersonic case is larger than in the hypersonic, which will result in a lower rate of compression. Moreover, in the hypersonic case, a secondary shock wave is produced past the body, probably as a result of the strong expansion in the forebody.

Studying the pressure plots, it can be seen that the major increase is obtained immediatly after the normal shock, and as the shocks starts to curve, the pressure gradient decreases. In the hypersonic case, these differences are higher.

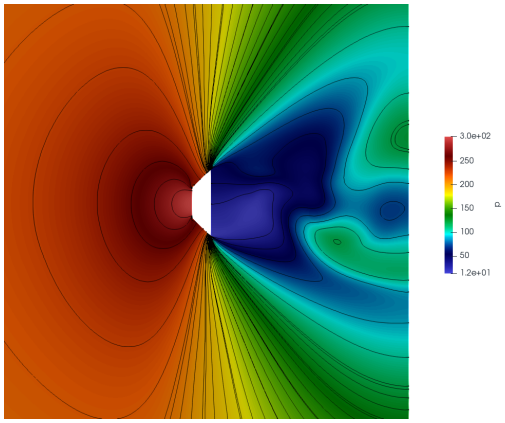
Regarding the Mach number, in both the supersonic and the hypersonic case the maximum value is barely even with the freestream. Between both phases, it is observed that in the hypersonic phase a secondary shock is created, because after the expansion the flow has been acclerated close to $M = 4$.

Finally, regarding the temperatures, the maximum value is around $T = 1400K$. These result indicates that the frontal part of the aeroshell should be made of a high-temperature material, probably ceramic. Despite this fact, these temperature results should be analysed in further design stages, when the structural design of the capsule is faced.

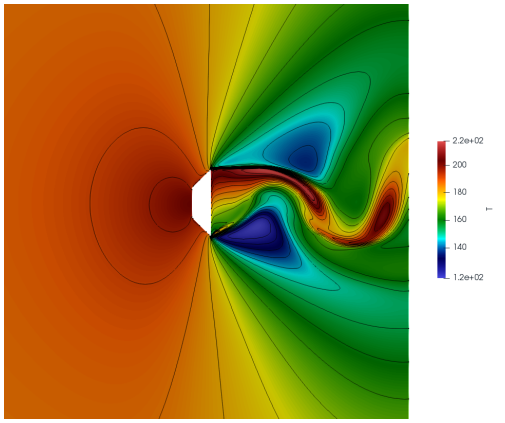
After presenting the results for the 70° aeroshell, the ones corresponding to the 45° aeroshell are plotted in Figure 3.27, Figure 3.28 and Figure 3.29. Its behaviour is practically identical to the 70° aeroshell, and therefore only the differentiating aspects are going to be discussed:



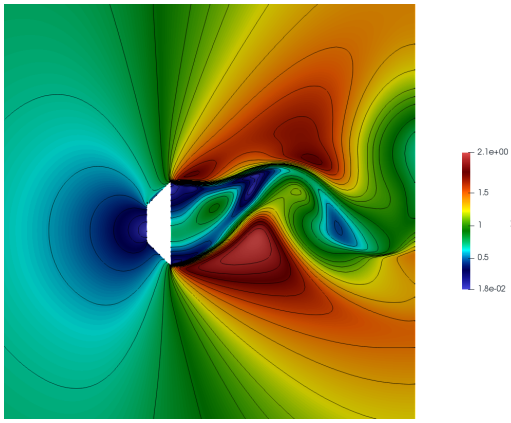
(a) Mach number, $\alpha = 2°$



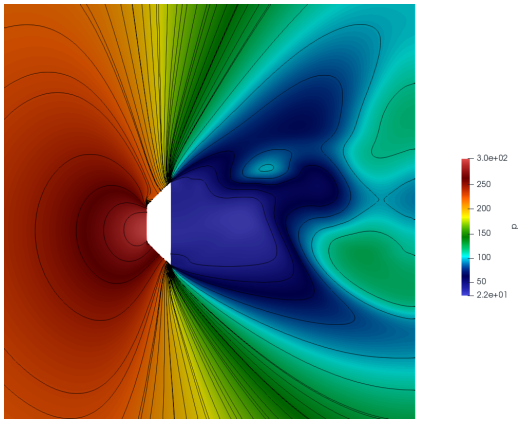
(b) Pressure, $\alpha = 2°$



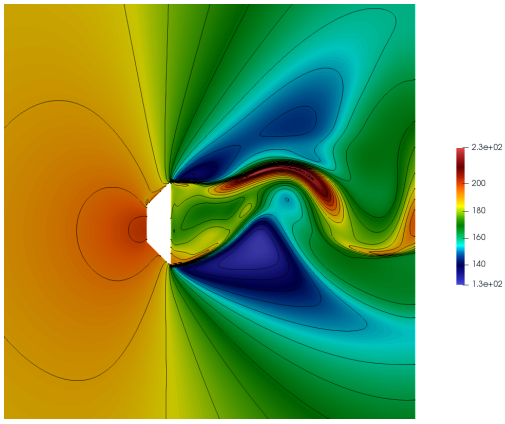
(c) Temperature, $\alpha = 2°$



(d) Mach number, $\alpha = 8°$

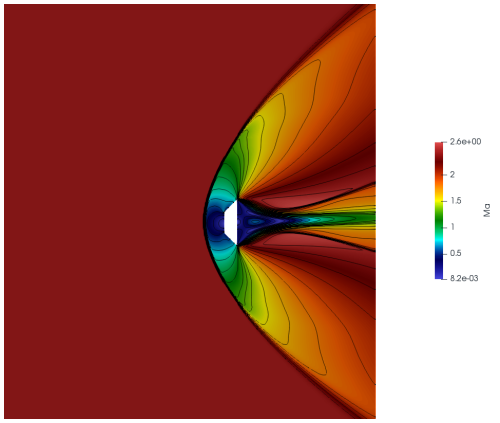


(e) Pressure, $\alpha = 8°$

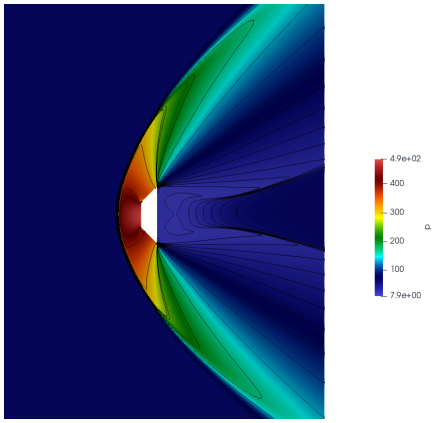


(f) Temperature, $\alpha = 8°$

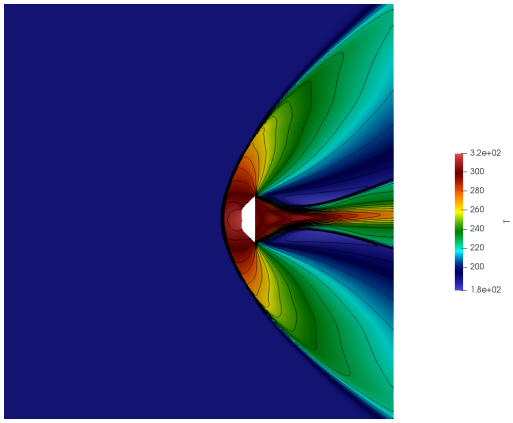
Figure 3.27: Results on temperature, pressure and Mach number for $M = 0.8$ and $D = 3.5m$



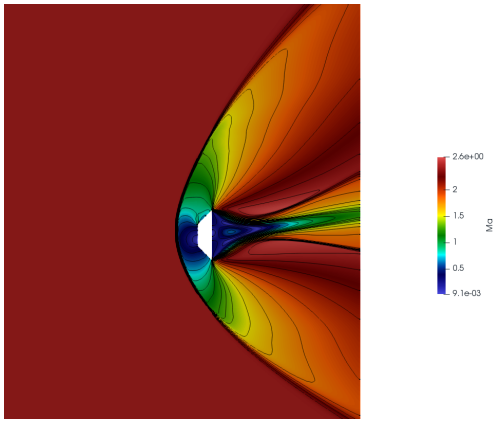
(a) Mach number, $\alpha = 2°$



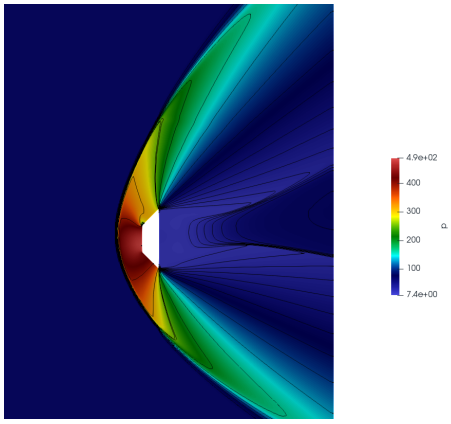
(b) Pressure, $\alpha = 2°$



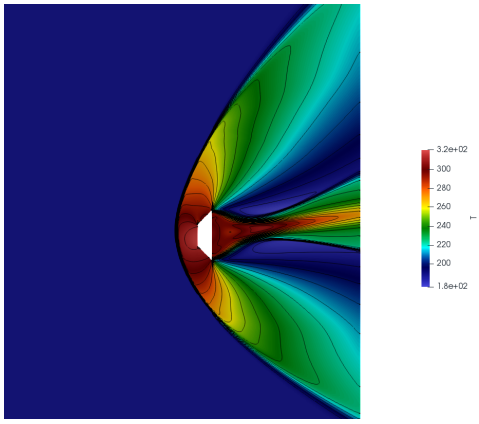
(c) Temperature, $\alpha = 2°$



(d) Mach number, $\alpha = 8°$

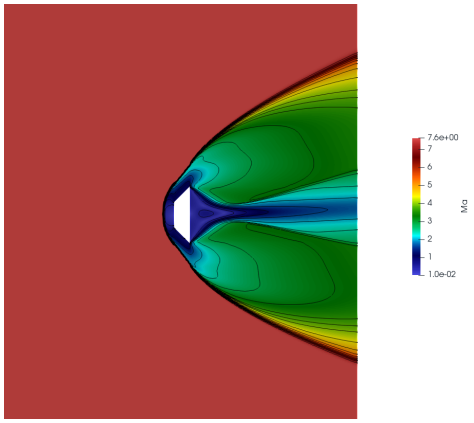


(e) Pressure, $\alpha = 8°$

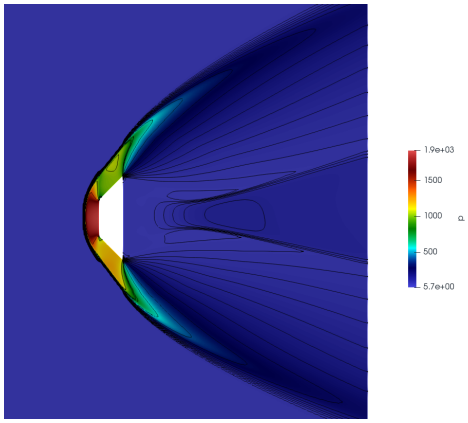


(f) Temperature, $\alpha = 8°$

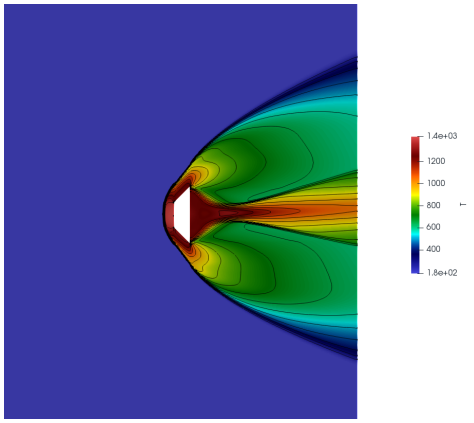
Figure 3.28: Results on temperature, pressure and Mach number for $M = 2.4$ and $D = 3.5m$



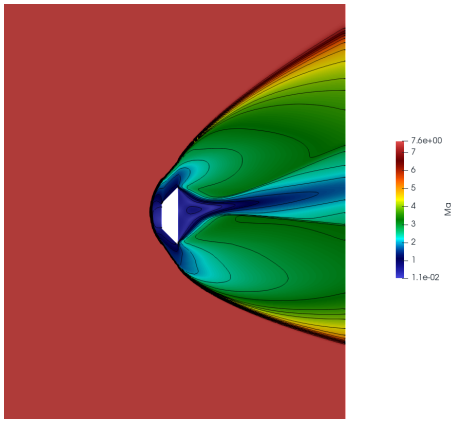
(a) Mach number, $\alpha = 2°$



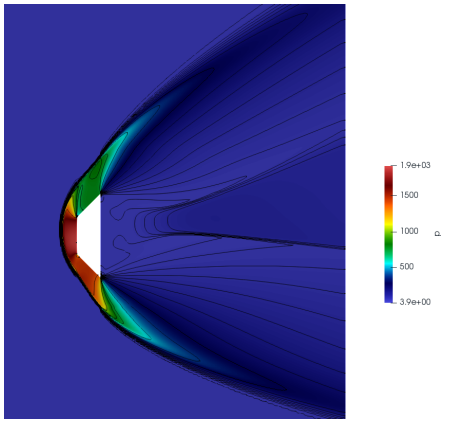
(b) Pressure, $\alpha = 2°$



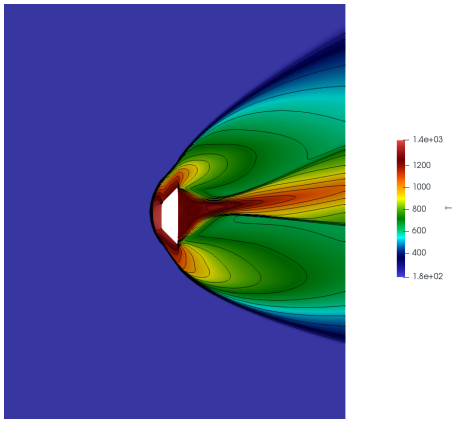
(c) Temperature, $\alpha = 2°$



(d) Mach number, $\alpha = 8°$



(e) Pressure, $\alpha = 8°$



(f) Temperature, $\alpha = 8°$

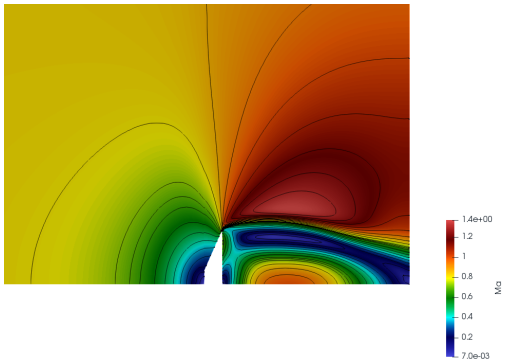
Figure 3.29: Results on temperature, pressure and Mach number for $M = 7.5$ and $D = 3.5m$

As mentioned above, in the transonic phase the solution presents a more oscillating wake. This is because of the reduced diameter of this geometry, which presents a wake that starts to show these vortexs earlier and, as the domain is the same for both cases, this behaviour is observed here and not in the 70° aeroshell. This fact should not alter the results, as for both cases there were non-reflective boundary conditions applied to their respective distances (at approximately 30 times the diameter of each aeroshell).

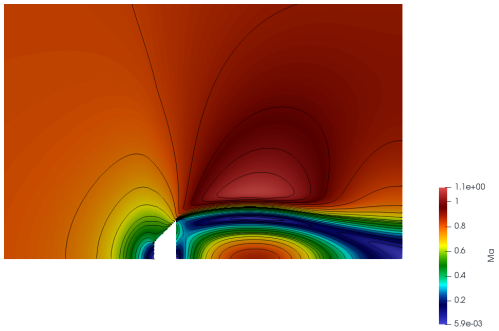
The solution for the supersonic and hypersonic cases is very similar to the previous geometry. The only difference is the behaviour of the isolines over the oblique face of the aeroshell. This occurs due to the different geometry, which will affect at the propagation of the shock wave. Due to this effect, a secondary shock wave is created over the oblique face of the aeroshell, making the deceleration even stronger. This effect can also be seen in the temperature peak which, in this case, is close to $2000K$. This effect should be studied further to determine if this can affect the capsule's integrity.

3.4.8.2 Axisymmetric domain

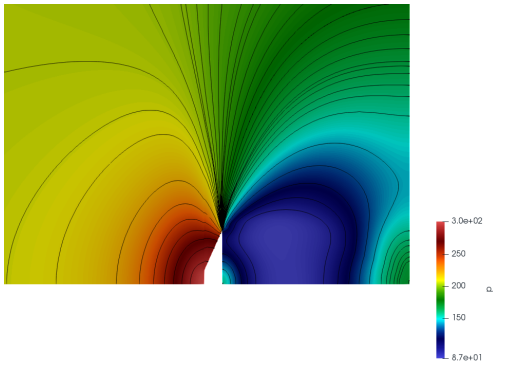
The results for the axisymmetric cases (at the same Mach number as the 2D cases) are presented in this section. The results obtained compared the 70° and 45° aeroshell at transonic, supersonic and hypersonic phases. Figure 3.30 compares the results for the transonic phase:



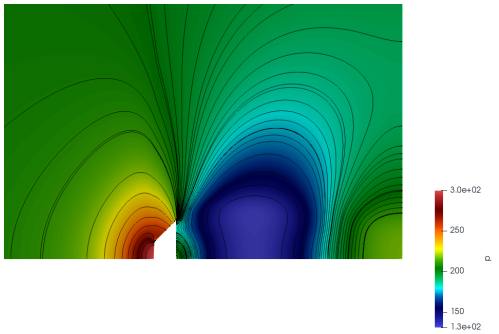
(a) Mach number, $D = 6m$



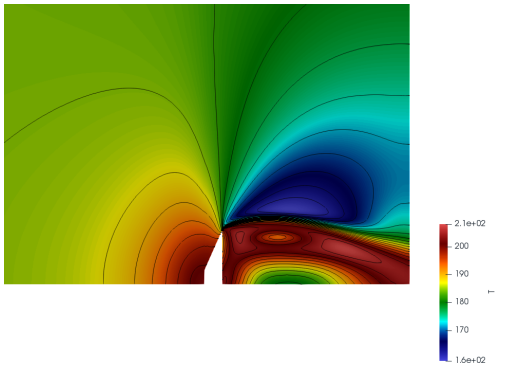
(b) Mach number, $D = 3.5m$



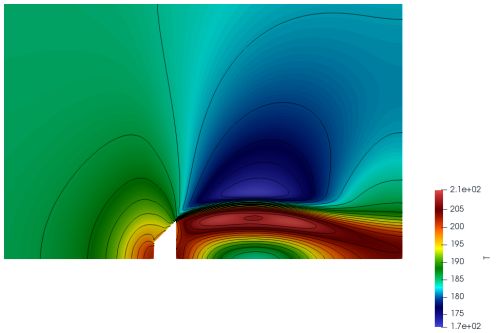
(c) Pressure , $D = 6m$



(d) Pressure , $D = 3.5m$



(e) Temperature , $D = 6m$



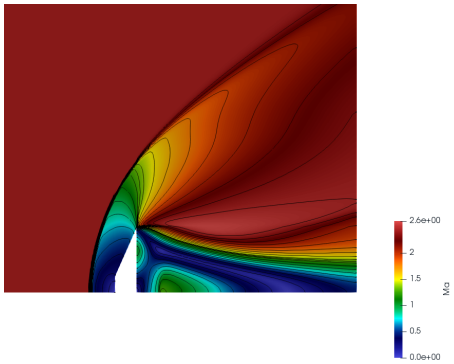
(f) Temperature , $D = 3.5m$

Figure 3.30: Results of temperature, pressure and Mach number for $M = 0.8$ for both geometries in axisymmetic flow

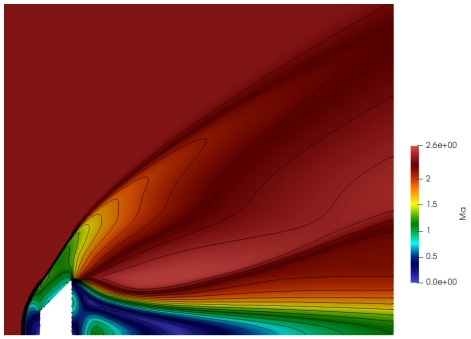
There are several differences compared with the 2D case. First of all, the increase in the Mach number is much lower. This is due to the 3D effects, which are taken into account here and not in the 2D case. Another difference is the aspect of the wake, which in this case presents a zone of high velocity which, again, could be due to the absence of viscous effects and should be further studied in future stages of the project.

Regarding the pressure, the isolines present a resembling behaviour to the 2D results in the forebody part. However, in the rear part the wake seems much more dependent on the aeroshell shape than in the 2D analysis. This effect accounts for the fact that, in axisymmetric analysis, the intensity of any aerodynamic effect is reduced due to 3D flow effects.

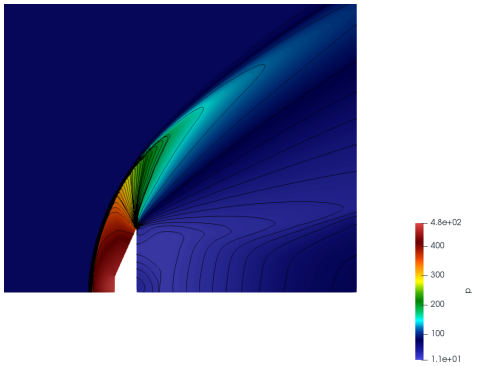
After the transonic results, Figure 3.31 and Figure 3.32 show the results for the supersonic and hypersonic phases, respectively:



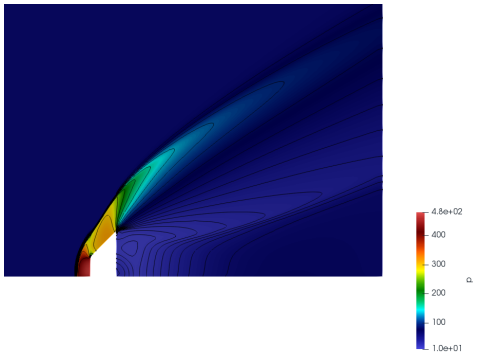
(a) Mach number, $D = 6m$



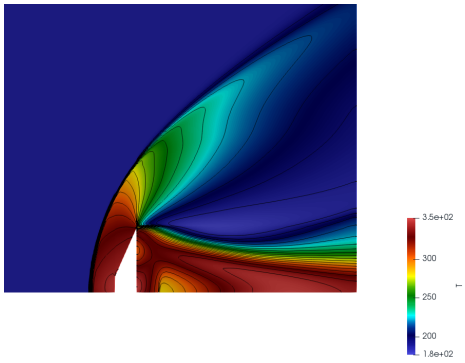
(b) Mach number, $D = 3.5m$



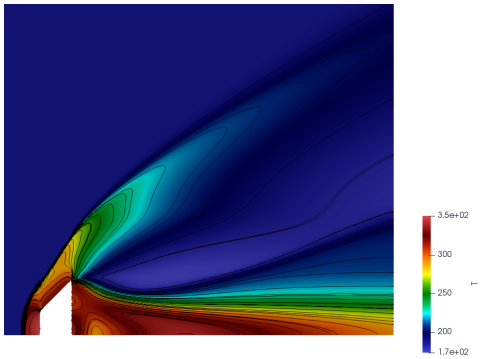
(c) Pressure , $D = 6m$



(d) Pressure , $D = 3.5m$

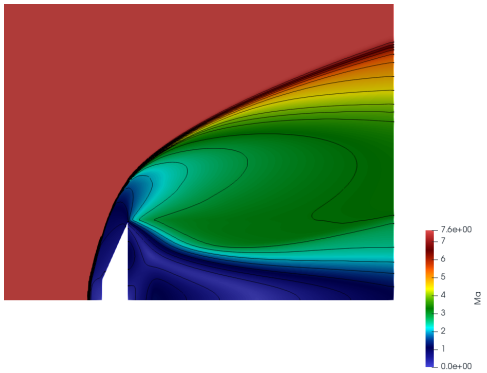


(e) Temperature , $D = 6m$

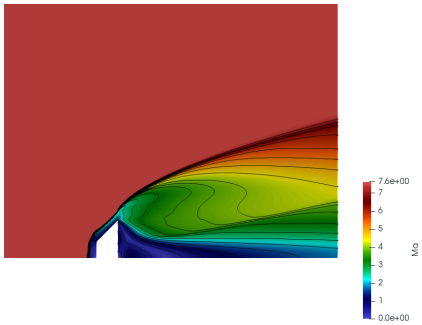


(f) Temperature , $D = 3.5m$

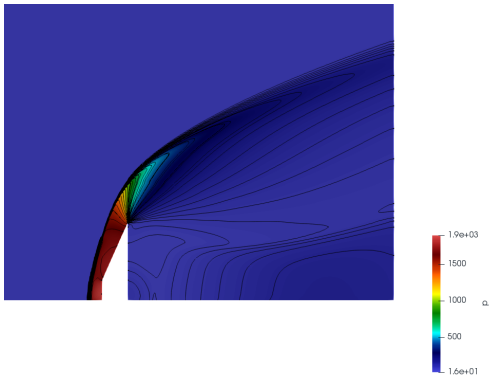
Figure 3.31: Results of temperature, pressure and Mach number for $M = 2.4$ for both geometries in axisymmetic flow



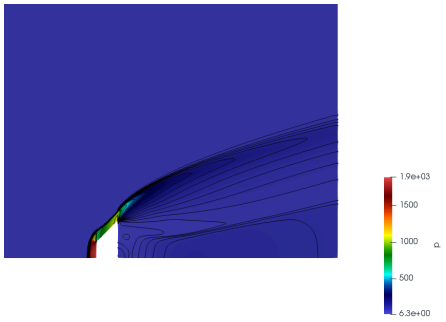
(a) Mach number, $D = 6m$



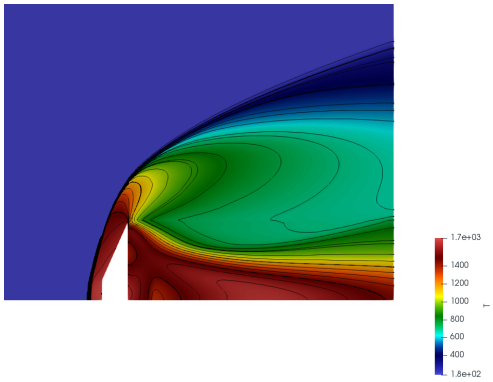
(b) Mach number, $D = 3.5m$



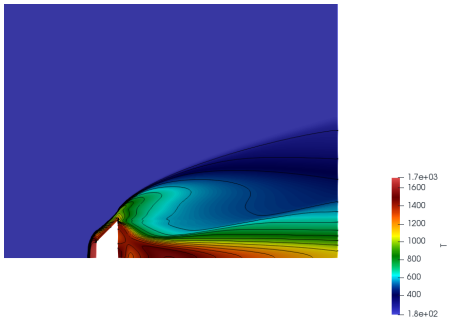
(c) Pressure , $D = 6m$



(d) Pressure , $D = 3.5m$



(e) Temperature , $D = 6m$



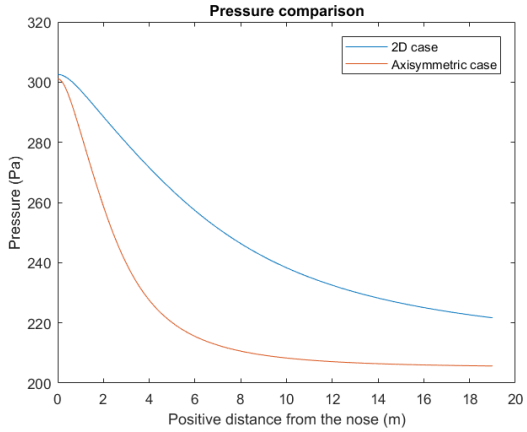
(f) Temperature , $D = 3.5m$

Figure 3.32: Results of temperature, pressure and Mach number for $M = 7.5$ for both geometries in axisymmetic flow

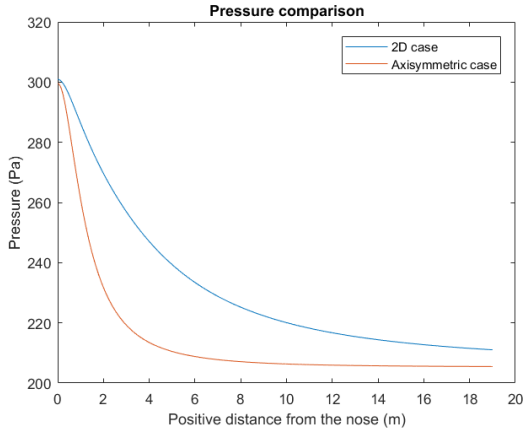
In these two phases, the difference with the 2D case is that the shock wave strength is much lower, as the shock cone is smaller. This, again, comes as a result of the 3D effects that the axisymmetric flow take into account. Comparing both geometries, it is also observed that the 45° aeroshell achieves a higher attachment of the shock wave (specially in its oblique face), which will be probably due to its geometry, which, as well as in the 2D case, could create a second shock wave after the primary one. Finally, regarding the temperatures, in this case the peak temperature has reduced for both geometries to $T = 1700K$.

3.4.8.3 Comparison of pressure distributions

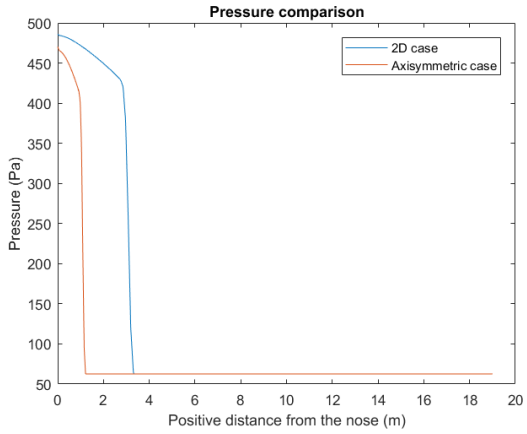
An important value to check the validity of these results is the pressure variation after the shockwave. It is expected that, in the 2D case, for all the regimes the pressure jump be higher value than in he axisymmetric case. Figure 3.33 shows this comparison for the transonic, supersonic and hypersonic case (the data is taken from a freestream point to the stagnation point):



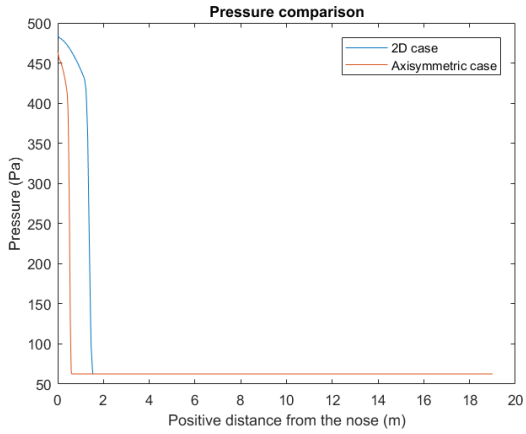
(a) Transonic, $D = 6m$



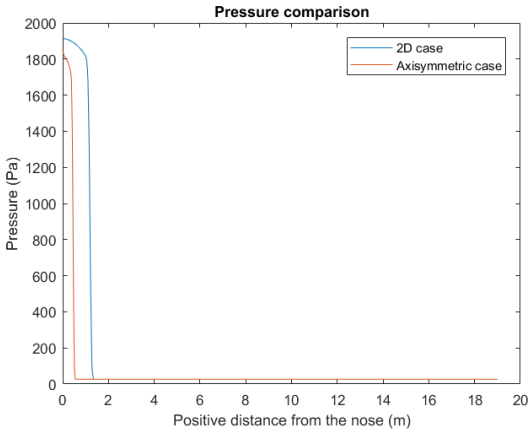
(b) Transonic, $D = 3.5m$



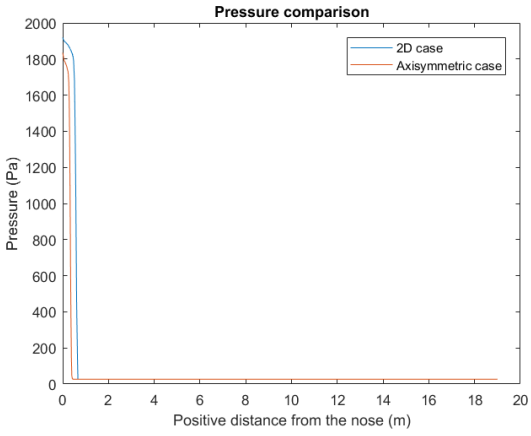
(c) Supersonic , $D = 6m$



(d) Supersonic , $D = 3.5m$



(e) Hypersonic, $D = 6m$



(f) Hypersonic , $D = 3.5m$

Figure 3.33: Results of temperature, pressure and Mach number for $M = 7.5$ for both geometries in axisymmetic flow

As expected, the pressure values achieved in the 2D case are higher than the ones achieved in the axisymmetric case. These differences grow slightly with the Mach number. Moreover, in the supersonic and hypersonic phase, the shock wave starts before in the 2D analysis than in the axisymmetric one. This is because of the higher intensity of the shock wave in the 2D case, which makes it to be more deattached from the geometry than in the axisymmetric case.

With all these data analysed, it can be concluded that the results obtained from the aerodynamic analysis are satisfactory and, besides they should be refined in further stages of the project (specially the transonic ones), they can be used in the final aerodynamic model.

3.5 Calculation of aerodynamic coefficients

In order to carry out the simulations, a suitable aerodynamic model has to be defined. Once the aerodynamic forces have been computed on the CFD analysis, it has been noticed that each force depends both on the angle of attack and the Mach number. The behaviour of the curves and its values are similar to the ones presented in (Edquist et al., 2019), which obtained aerodynamic data for geometries similars to the ones studied. It is important to note that, due to the oscillating behaviour of the transonic data, the adopted model is a mean of the obtained results.

The procedure undergone to obtain the aerodynamic model has been the following. First of all, the coefficient results have been analysed only as a function of the AoA. Both geometries have shown that:

- The C_L and C_M functions behave as a first degree polynomial
- The C_D function behave as a second degree polynomial

All these models have shown a good accuracy in the different Mach numbers. Therefore, each of these functions have been computed as:

$$\begin{cases} C_D(M,\alpha) = C_{D0}(M) + C_{D\alpha}(M)\alpha + C_{D\alpha^2}(M)\alpha^2 \\ C_L(M,\alpha) = C_{L\alpha}(M)\alpha \\ C_M(M,\alpha) = C_{M\alpha}(M)\alpha \end{cases} \tag{58}$$

Equation 58 shows that each coefficient depend on the Mach number also. In order to adjust these dependency, each coefficient of C_D, C_L and C_M have been plotted vs. M and a polynomial have been adjusted (from first to sixth degree). The coefficients have been divided, in the lift-drag-moment case, into transonic to low supersonic, for $0.75 < M < 2$, and high supersonic to hypersonic, for $M > 2$ and in the ballistic case, they have been divided into transonic, $0.75 < M < 1.5$, supersonic, $1.5 < M < 5$ and hypersonic, $M > 5$:

Lift-drag-moment case

$$0.75 < M < 2 : \begin{cases} C_D(M,\alpha) = C_{D0}(f(M^4)) + C_{D\alpha}(f(M))\alpha + C_{D\alpha^2}(f(M^3))\alpha^2 \\ C_L(M,\alpha) = C_{L\alpha}(f(M^2))\alpha \\ C_M(M,\alpha) = C_{M\alpha}(f(M^3))\alpha \end{cases} \tag{59}$$

$$M > 2 : \begin{cases} C_D(M,\alpha) = C_{D0}(f(M^4)) + C_{D\alpha}(f(M^6))\alpha + C_{D\alpha^2}(f(M^3))\alpha^2 \\ C_L(M,\alpha) = C_{L\alpha}(f(M^4))\alpha \\ C_M(M,\alpha) = C_{M\alpha}(f(M^6))\alpha \end{cases} \tag{60}$$

Ballistic case

$$0.75 < M < 1.5 : C_D(M,\alpha) = C_{D0}(f(M)) \tag{61}$$

$$1.5 < M < 5 : C_D(M,\alpha) = C_{D0}(f(M)) \tag{62}$$

$$M > 5 : C_D(M,\alpha) = C_{D0}(f(M^2)) \tag{63}$$

In these nine euqations, $C_i f(M^j)$ represents that the coefficient C_i is of the form of a pollynomial that has a degree j and depends on M. Two different models have been developed, one for each geometry. The exact values of each equation can be consulted on Appendix F.

3.6 Selection of the parachute decelerator

In this section the parachute deceleration system will be designed. Several assumptions have been done:

- The flow and geometry during the inflation process is supposed to be axisymmetric.
- The descent is ballistic.
- The snatch force at the beginning of the inflation process is neglected, only the drag created by the volume of the bag before the snatch will be considered.
- The canopy weight is supposed to be a 15% of the system's weight.
- The inflation before the line strech is a percentatge of the inflation at bag strip, which will be considered a 10%.
- The system operates under the finite mass condition.

- The maximum landing velocity permitted is $25\frac{m}{s}$.

An existent model has been selected according to the landing velocity restrictions of (Pasolini et al., 2017). After that, the inflation phase have been defined, using the semi-empirical models presented in (Knacke, 1991), (Solt and Ria, 1961) and (Lingard, 1995). Finally, the characteristic curve of the parachute's drag force is presented.

3.6.1 Parachute Model

The selected model is the parachute used in the Viking mission (see the experimental model in (Moog, 1973). This parachute is depicted in Figure 3.34, and it would let a terminal velocity of $17,5\frac{m}{s}$ if the system drag coefficient was $C_D = 1$ (Pasolini et al., 2017). It will be deployed at a $M = 0.75$.

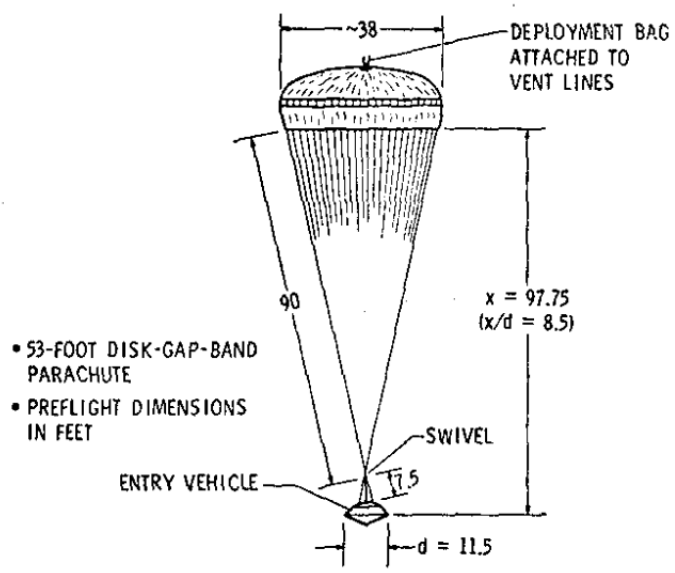


Figure 3.34: Selected parachute, in feets (source (Moog, 1973))

The parachute will be supposed to have a standard porosity. The inflation phase will include a 10% inflation at bag strip, which stands for the porosity supposition.

3.6.2 Inflation phase

The inflation presents the following stages, shown in Figure 3.35. First of all, the canopy is opened and starts to expand as the air starts to inflate it. This air flux inflates the canopy's crown and stretches the lines of the decelerators, stage where the snatch force is produced. Finally, the canopy reaches its fully inflation stage, and keeps over inflating

(drag peak) until the surrounding mass of air deforms this shape, thus reshaping the canopy to its stationary shape (Knacke, 1991).

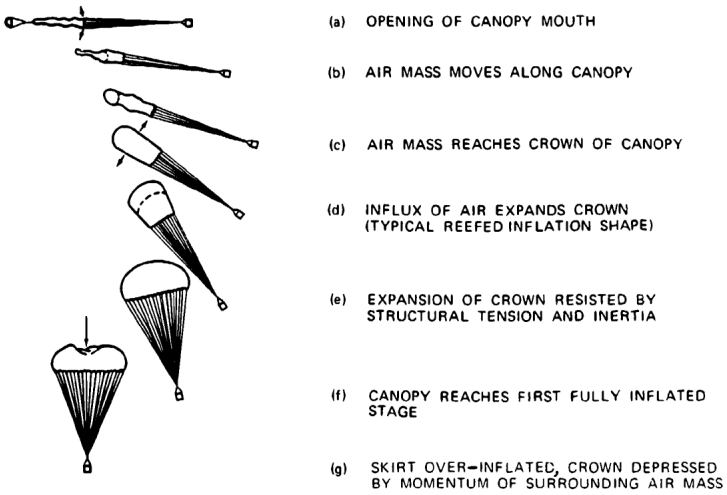


Figure 3.35: Stages of the inflation process (source (Knacke, 1991))

In order to design the inflation phase, several parameters have to be defined: the inflation time, which will depend on the snatch velocity, and the axial force, which will depend on the mentioned inflation time.

3.6.2.1 Dynamics during inflation process

The system of forces that act over the parachute during its inflation can be seen in Figure 3.36. The parachute will be supposed to open axisymetrically, which means that no lift is produced and only an axial force is taken into account.

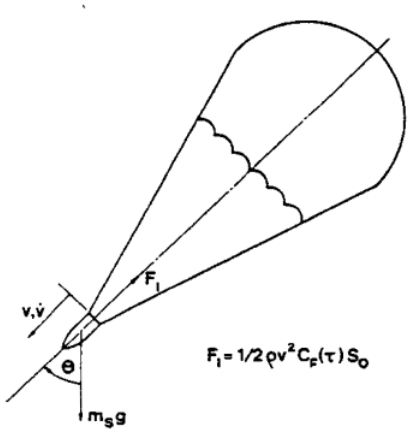


Figure 3.36: System of forces during parachute inflation (source (Lingard, 1995))

This system of forces can be written as follows (Lingard, 1984):

$$m_s \frac{dV}{dt} = m_s g \cos\theta - \frac{1}{2}\rho V^2 S C_D - a p_{mass} - V \frac{d a p_{mass}}{dt} \tag{64}$$

$$\frac{d\theta}{dt} = -\frac{g \sin\theta}{V} \tag{65}$$

where SC_D is the instantaneous drag area and depends on the time. Note that this model includes the added mass term, which must take into account the derivative of α_{11}. This depends on the diameter and changes during the inflation.

In Equation 64, the drag area SC_D must be calculated. To this end, the following model is adopted ((Ludtke, 1972) and (Knacke, 1991)):

$$\frac{SC_D}{(SC_D)_0} = \begin{cases} C_x \left((1-\eta) \left(\frac{t}{t_i}\right)^3 + \eta \right)^2, & \text{if } t < t_i \\ a \left(\frac{t}{t_i}\right)^2 + b \left(\frac{t}{t_i}\right) + c, & \text{if } t < (1 + t_{irat}) t_i \\ 1, & \text{if } t > (1 + t_{irat}) t_i \end{cases} \tag{66}$$

where t_{irat} is the filling time percentatge that the parachtue needs to reduce its opening shock force to the steady-state. The data to compute this value is found in (Ludtke, 1972). Moreover, C_x is the opening-force scale factor and for a disk-gap-band parachute, $C_x = 1.3$ (Knacke, 1991). This scale factor is intended only for infinite mass conditions, but in this case agrees reasonably well with the experimental results.

This model is suitable either or finite or infinite mass parachutes. It also considers the initial filling of the canopy before the line stretch, and comprises the time between the line stretch and the full opening of the parachute.

In this case, and according to (Knacke, 1991) and the Viking's results (Moog, 1973), the inflation time t_i can be calculated as

$$t_i = \frac{0.65 \lambda_G D_0}{V_{sn}} \tag{67}$$

and where, in this case, the geometrical porosity of the canopy λ_G is equivalent to the 12.5% of the projected canopy surface. Equation 67 agrees with the experimental data of the tests presented in (Moog, 1973).

Finally, the only parameter left to calculate the t_i is the snatch velocity V_{sn}.

3.6.2.2 Snatch velocity

In this case, the velocity at parachute's opening not variable, thus, an estimation of the snatch velocity can be done. This velocity is the velocity of the parachute when line stretch occurs. When the parachute is deployed, the canopy is accelerated from the payload. During this process, the lines of the parachute gain enery, thus returning the parachute towards its deployment position. This elastic energy will produce a peak known as snatch force. This point is assumed to be the start of the parachute filling (Knacke, 1991).

In order to calculate the snatch force and velocity, the equations of energy and velocity are derived, respectively. In this section a general view in order to calculate the snatch velocity is presented. The whole process can be found in Appendix E.

First of all, and using the expressions in the former appendix (extracted from (Solt and Ria, 1961)), the J parameters are calculated. These parameters are the ratio between the drag area and the volume of the system:

$$\begin{cases} J_1 = \frac{\rho (C_D S)_b}{2(m_b + m_c)} \\ J_b = \frac{\rho (C_D S)_b}{2 m_b} \\ J_c = \frac{\rho (C_D S)_c}{2 m_c} \end{cases} \tag{68}$$

After these initial calculations, the following set of equations is obtained in order to calculate v_d, or velocity at line extension:

$$\begin{cases} V_d = \frac{V_0}{J_1 V_0 t_1 + 1} \\ L_1 = \sqrt{\frac{g^2 t_1^4}{4} + \left(V_0 t_1 - \frac{1}{J_1} \ln 1 + J_1 V_0 {t_1}^2 \right)} \end{cases} \tag{69}$$

V_0 is the velocity of the capsule at the parachute's deployment, which occurs at $M = 0.75$. With that value, and using the equation of static line length L_1, the time of lines elongation t_1 can be calculated.

With these results, the time to separate the primary and secondary bodies t_2 is calculated using the definition of the length of suspension line L_{sn}.

$$\begin{cases} L_{sn} = V_d t_2 - \frac{1}{J_c} \ln 1 + J_c V_d t_2 \\ V_{sn} = \frac{V_d}{J_c V_d t_2 + 1} \end{cases} \tag{70}$$

With all these data, the snatch velocity yields a value of $112.1552 \frac{m}{s}$.

3.6.3 Evolution of drag force during inflation

With the snatch force, it is possible to calculate the inflation time using Equation 67 which in this case yields $2.4s$. This time interval agrees with the data presented in (Moog, 1973).

Finally, the inflation curve is plotted, using Equation 66. The drag evolution is plotted vs. the normalised time:

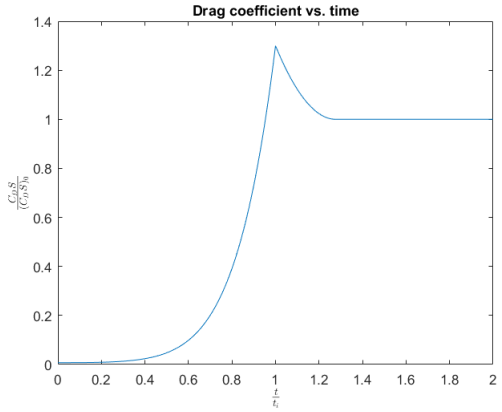


Figure 3.37: Drag ratio vs. dimensionless time during inflation

It should be noted that this inflation curves assumes that the transition between the peak force and the steady state force is instantaneous. However, in the simulator, when the inflation time is over the parachute drag evolution will be determined by the experimental drag force obtained in the Viking tests in (Moog, 1973) and shown in Figure 3.38:

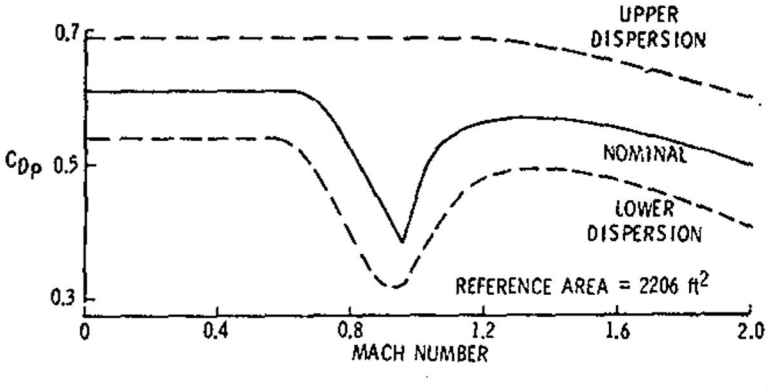


Figure 3.38: Drag coefficient vs.drag functions (source *Moog et al.*)

The function that is going to be used is the nominal one. This function is the mean of the measured data in the four tests on the Viking parachute. An analytical function $C_D = f(M^6)$ has been adjusted to the experimental data. The drag coefficient function implemented in the simulator is presented in Appendix F.

Chapter 4

Flight simulator analysis

After developing the algorithm for the flight simulator and calculating the aerodynamic parameters, the results of the EDL sequence are presented. Four analysis are carried out:

- 70-degree cone aeroshell: two analysis, one with only drag force (ballistic descent) and another with lift, drag and pitching moment.
- 45-degree cone aeroshell: two analysis, one with only drag force (ballistic descent) and another with lift, drag and pitching moment.

In both cases, the parachute will descent in a ballistic state.

4.1 Simulation results

After introducing the aerodynamic models into the flight simulator, the EDL sequence results are presented. Several assumptions have been taken into account:

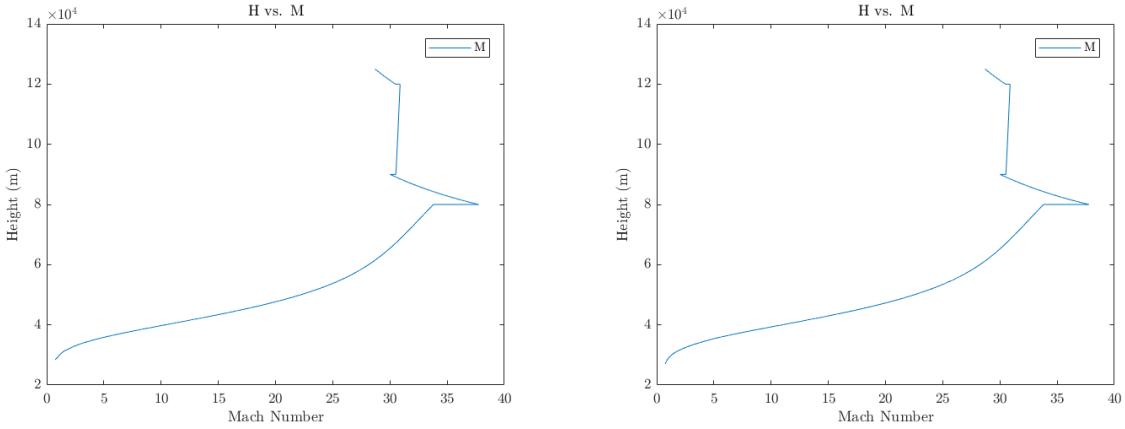
- The reentry flight path angle have been supposed to be -13, which performs good in Mars reentries (Pasolini et al., 2017).
- The parachute stability has been calculated with the payload center of gravity as the reference point for moments.

In the next sections, the hypersonic phase results are presented. One of the two presented geometries will be selected due to its performance. After that, several parachute drops will be compared, changing the parachute's diameter, yielding to the selection of a final configuration.

4.1.1 Hypersonic phase results

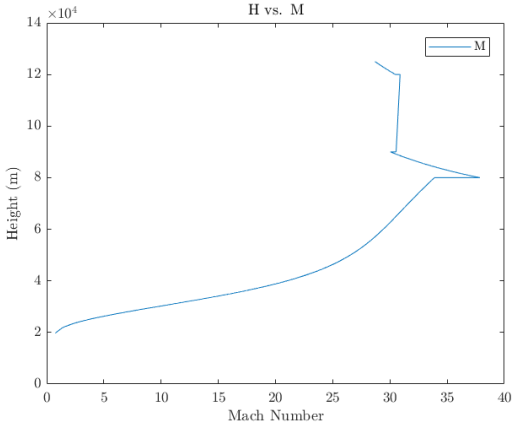
Two geometries at two different descent regimes (2D and ballistic) have been analysed. The following data has been analysed: the angle of attack α, the flight path angle γ, the Mach number M and the descent velocity V. All are plotted vs. the height H.

Figure 4.1 there is a comparison on the Mach number of the four descents (the axisymmetric and 2D descents of both geometries) in the hypersonic phase and in Figure 4.2 there is a comparision of the descent velocity.

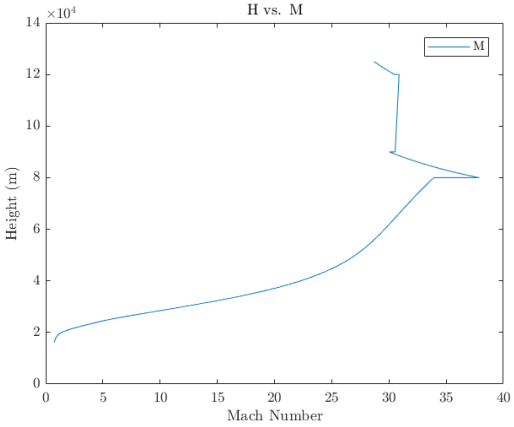


(a) D = 6 m, 2D

(b) D = 6 m, ballistic

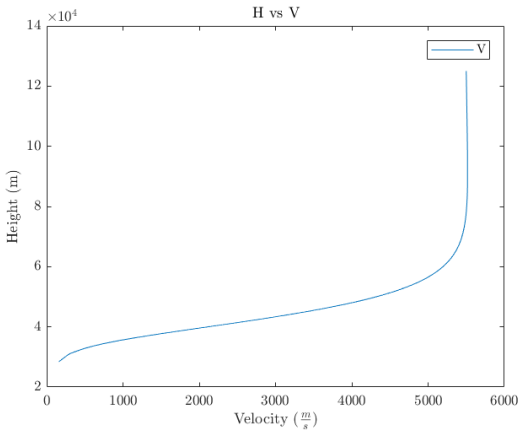


(c) D = 3 m, 2D

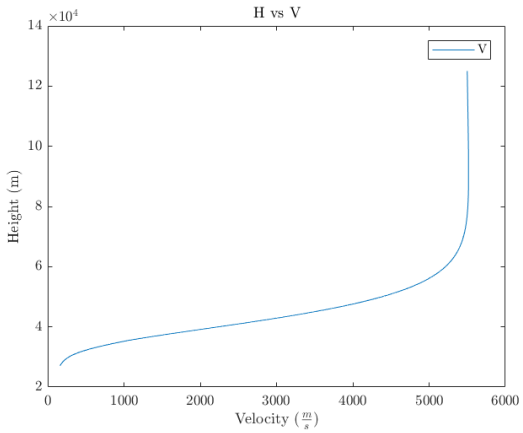


(d) D = 3 m, ballistic

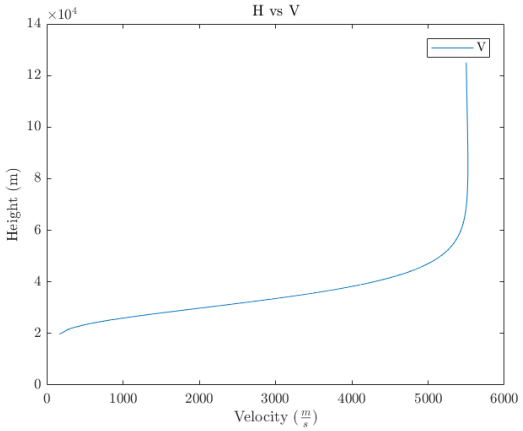
Figure 4.1: Altitude evolution vs. the Mach number



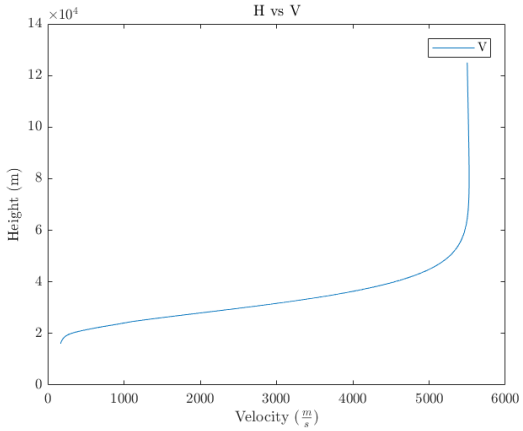
(a) D = 6 m, 2D



(b) D = 6 m, ballistic



(c) D = 3.5 m, 2D



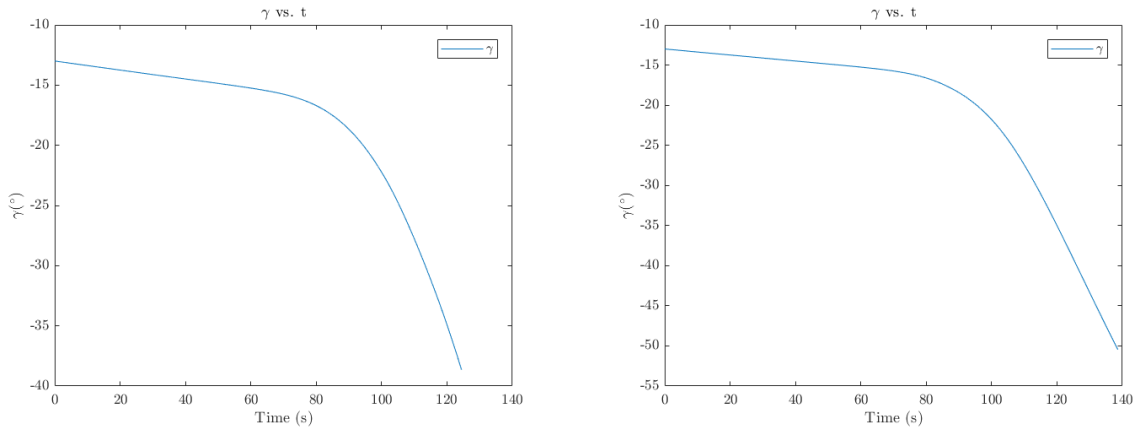
(d) D = 3.5 m, ballistic

Figure 4.2: Altitude evolution vs. the velocity

As it can be seen, the behaviour of all the descents is practically the same. The major difference is that in the ballistic cases, the parachute deployment height is slightly lower. This result is expected, as in the ballistic case there is no lift and also, as being a 3D approximation, the drag achieved is lower, as the shock wave has much less energy. Looking at the curve's behaviour in Figure 4.1, there appear some apparent discontinuities in the tendency of the function. This is due to the temperature distribution in Mars, which can vary drastically from one height to another (from 90000 meters to 80000 metres, for example). Finally, analysing the velocities distribution in Figure 4.2, the velocity keeps in the initial value (and even grows a bit) until reaching approximately 50000 m, where the real deceleration starts. This is explained due to the low density of Mars, which is practically zero until the former altitude.

From this obtained performances, it has been decided to opt for the **3.5 meters diameter** aeroshell (the one corresponding to 45°), as it offers practically the same performance as the 6 meter diameter, with smaller dimensions.

Finally, it is only necessary to analyse the attitude of the capsule. This is important for two reasons: first of all, if the capsule presented oscillations the design would be unacceptable, because it will end in high structural damage. Secondly, it is important to check the AoA range, because the aerodynamic model was calculated from 0° to 8°. A damping is going to be used to control possible oscillations. Figure 4.3 shows the evolution of the flight path angle:



(a) D = 3 m, 2D

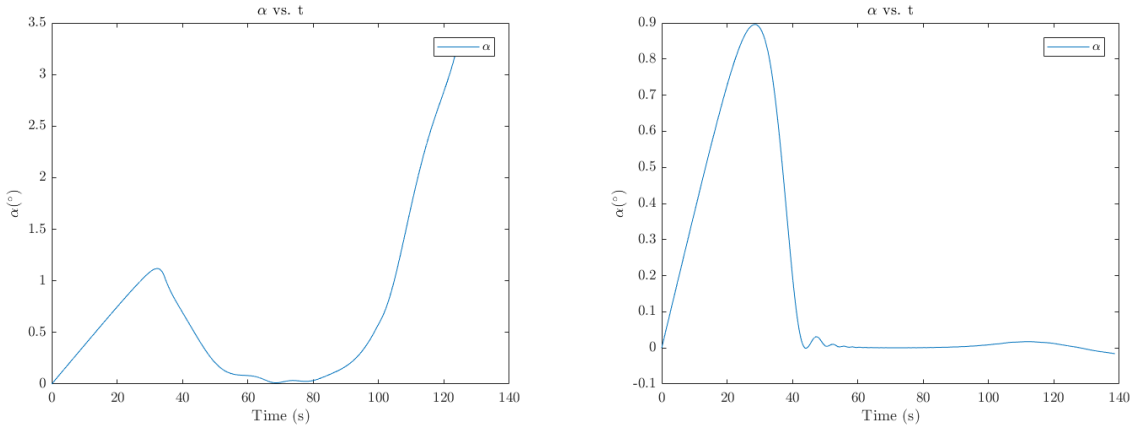
(b) D = 3 m, ballistic

Figure 4.3: Evolution of the flight path vs the time

As it can be observed, the results present no oscillations. This is because of the damping, which will be comented at the end of the section. Secondly, the flight path on the ballistic case is approximately 10° higher than in the 2D case. This can possibly be due to the effect of the lift force, which can make the angle of the capsule different from 0

in the trimming point.

Regarding the angle of attack, Figure 4.4 compares the evolution of α during the descent:



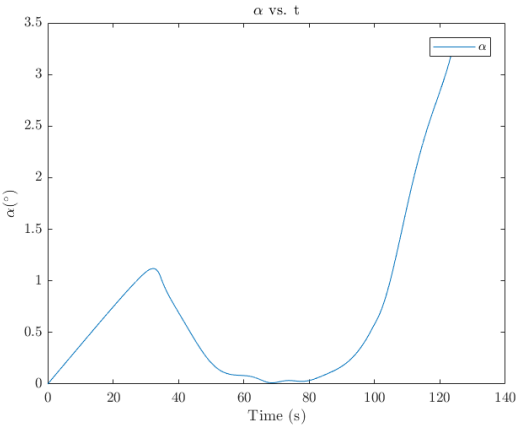
(a) D = 3 m, 2D

(b) D = 3 m, ballistic

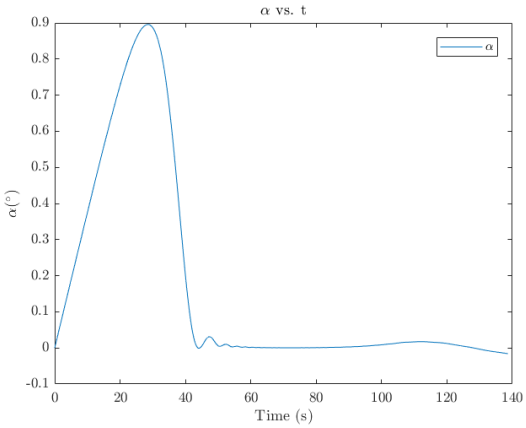
Figure 4.4: Evolution of the AoA vs the time

As it can be seen, the distribution of both cases is completely different, which could explain the effect observed in Figure 4.3. The evolution of the angle of attack in the 2D case presents a sudden peak around 100 seconds. This is because the aeroshell enters in the transonic zone, and the moment coefficient moment showed instabilities in such zone ($C_{m\alpha} > 0$ in certain zones. This can be because of the position of the centre of pressure in the transonic case, which in some cases was not the expected). In the ballistic case, however, the moment is taken simply as the contribution of the axial force which, all together with the damping, converges to 0 at the trimming point, as expected.

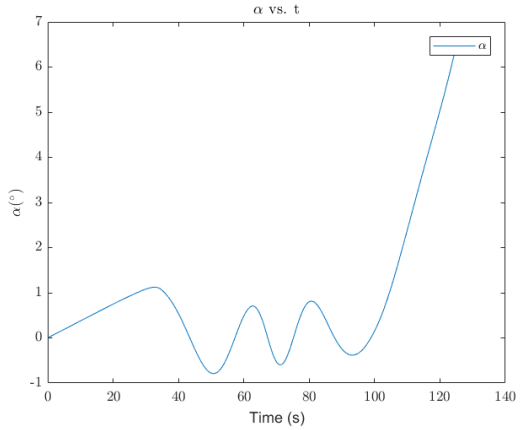
These results on the angle of attack have been damped with a damping coeffcient $C_{mq} = -0.5(\frac{V_\infty}{V_{reentry}})^3$, value which has been determined experimentally. In Figure 4.5 there is a comparison of the damped results and the undamped ones:



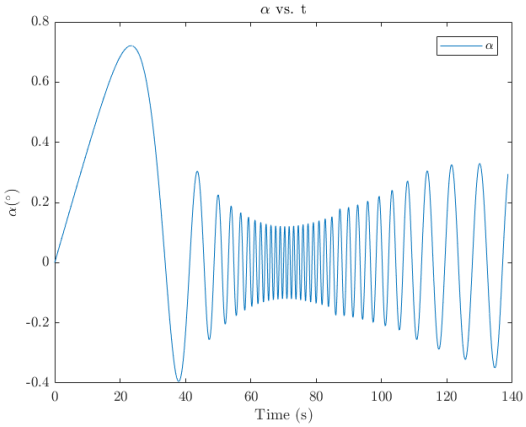
(a) D = 3 m, 2D with damping



(b) D = 3 m, ballistic with damping



(c) D = 3 m, 2D without damping



(d) D = 3 m, ballistic without damping

Figure 4.5: Comparison of damped and undamped results

The results show that the oscillations in the ballistic case are sever and unnaceptable on a capsule design. In the 2D case, although the oscillations are much lower, with the introduction of a damping coefficient the final AoA is almost half the one without damping, a behaviour that is desirable.

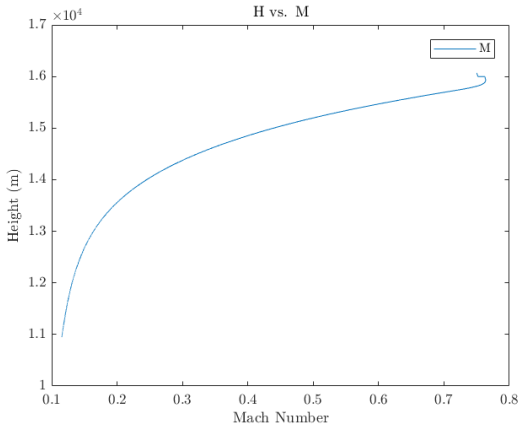
According to the obtained results, the contribution of the lift is important as it increases the parachute deployment altitude. Despite this fact, a 3D analysis should be done in order to completely characterize the effect of the lift force in the descent.

After the hypersonic descent phase, the subsonic is presented presented. The deployment of the parachute is $16068m$, which is the corresponding to the ballistic descent of the 45° cone. It has been choosen over the 2D case as it is a more realistic result (it is a better approximation to the real flow).

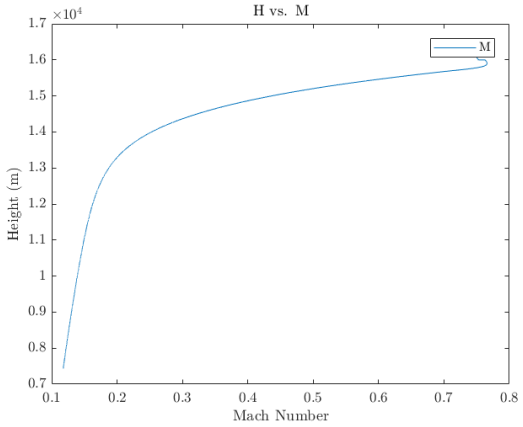
4.1.2 Subsonic phase results

The subsonic descent of he system is studied here. Three different configurations are going to be presented, and one final configuration is selected depending on the final altitude. The drops have been done at two initial θ_p angles, 5° and 30°. Finally, a comparison between the damped and undamped results of the final configuration is presented.

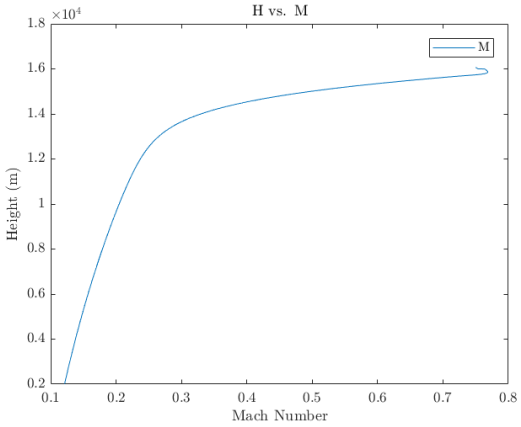
The three different parachutes that have been analysed have the same dimensions in the payload and the suspension lines, the only part that differs is the canopy diameter, which will be $D = [16m, 12m, 8m]$. Therefore, in Figure 4.6 there is a comparison in the Mach number and in 4.7 is plotted a comparison between the different descent velocities. Both figures are plotted at $\theta_{p0} = 30°$:



(a) D = 16 m, $\theta_p = 30°$

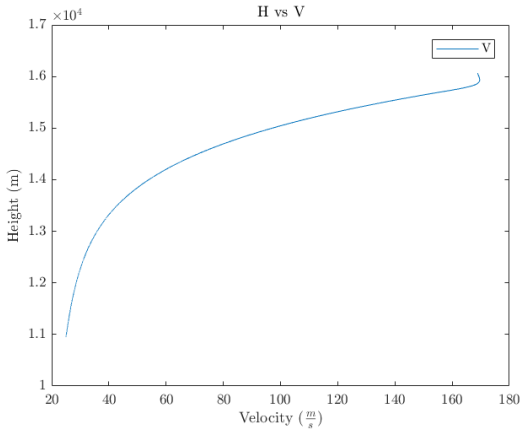


(b) D = 12 m, $\theta_p = 30°$

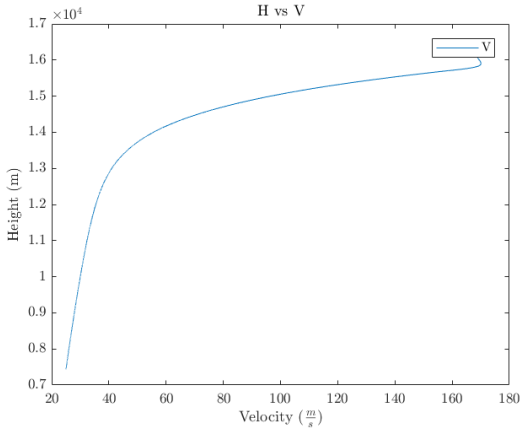


(c) D = 8 m, $\theta_p = 30°$

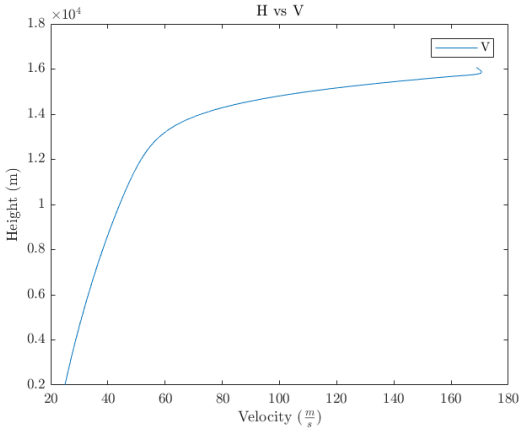
Figure 4.6: Height evolution vs. the Mach number



(a) D = 16 m, $\theta_p = 30°$



(b) D = 12 m, $\theta_p = 30°$



(c) D = 8 m, $\theta_p = 30°$

Figure 4.7: Height evolution vs. the velocity

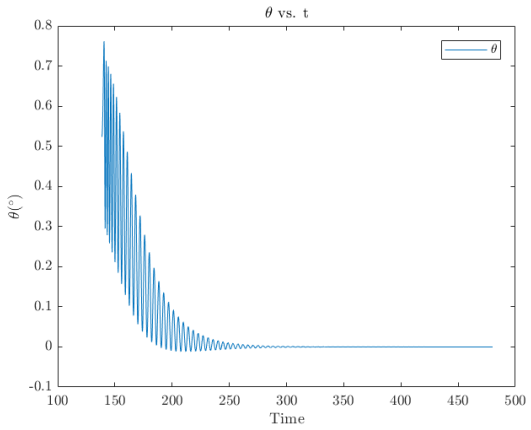
The results show that the velocity and the Mach number present the same behaviour. As it was expected, the landing height (or the height when $V_{term} = 25\frac{m}{s}$ is different for each case, yielding:

	D = 16 m	D = 12 m	D = 8 m
Height at V_{term} (m)	10948	7433	2009

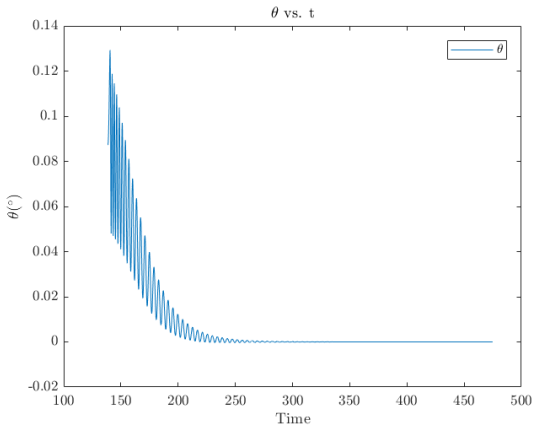
Table 4.1: Height at terminal velocity

As observed, the parachute, at its maximum diameter, needs approximately $6000m$ to reach the desired terminal velocity, which seems realistic. For this geometry, the best parachute would be the one with $D = 8m$, as there is no need of a larger diameter in order to achieve the landing (the landing region of interest in (Pasolini et al., 2017) ranges from $0m$ to $2000m$).

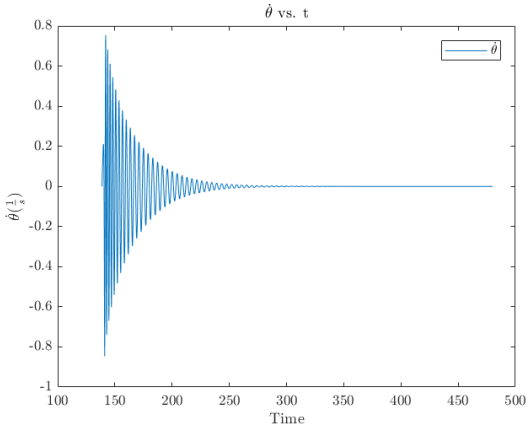
Finally, its important to analyse the parachutes dynamic stability behaviour. In Figure 4.8 there are plotted the results of pitch angle and pitch velocity for $D = 8m$. In this case, the results of both $\theta_{p0} = 5°$ and $\theta_{p0} = 30°$ are going to be presented, as their impact in the parachute's attitude is relevant.



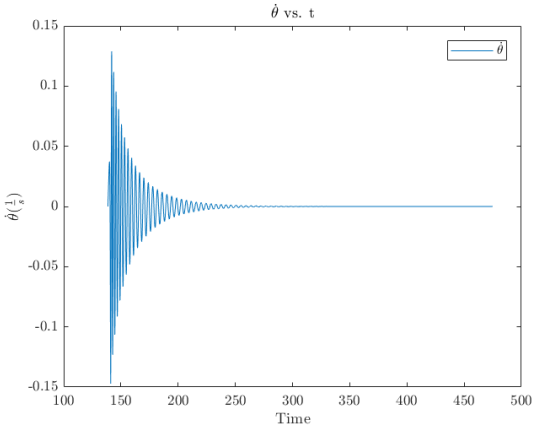
(a) θ evolution at $\theta_{p0} = 30°$



(b) θ evolution at $\theta_{p0} = 5°$



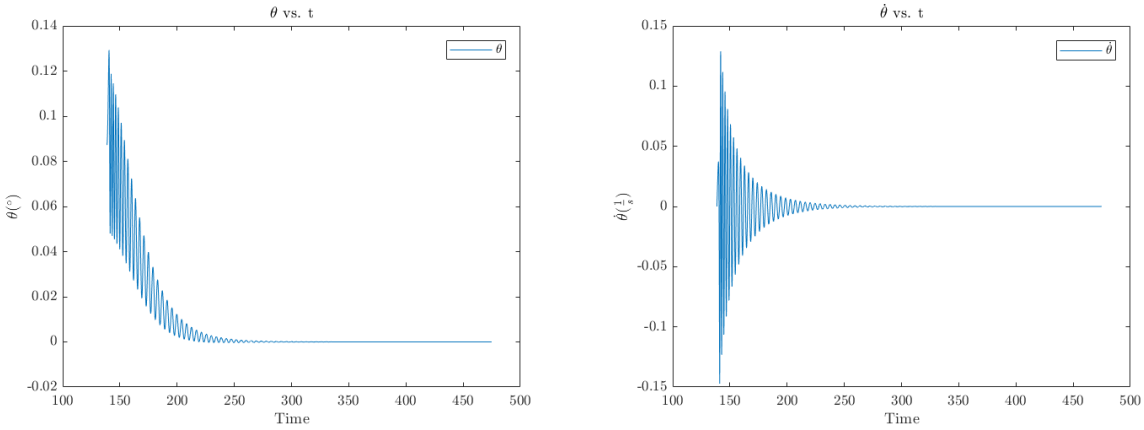
(c) $\dot{\theta}$ evolution at $\theta_{p0} = 30°$



(d) $\dot{\theta}$ evolution at $\theta_{p0} = 5°$

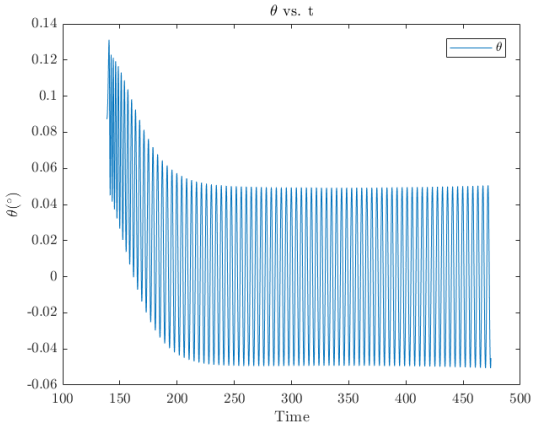
Figure 4.8: θ and $\dot{\theta}$ evolution vs. time

The results presented in Figure 4.8 behave as expected, as they rapidly stabilize at the trimming position, which for a ballistic descent is 0° for θ and $0\frac{rad}{s}$ for $\dot{\theta}$. The damping coefficient $C_{mq} = -0.075$ and has been determined experimentally. In Figure 4.9 there is a comparison between damped and undamped results for $\theta_{p0} = 5°$ (at $\theta_{p0} = 30°$ the parachute was too unstable and the simulation diverged).

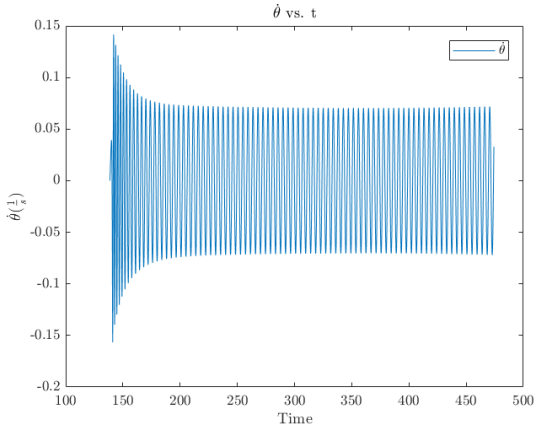


(a) θ evolution with damping

(b) $\dot{\theta}$ with damping



(c) θ evolution without damping



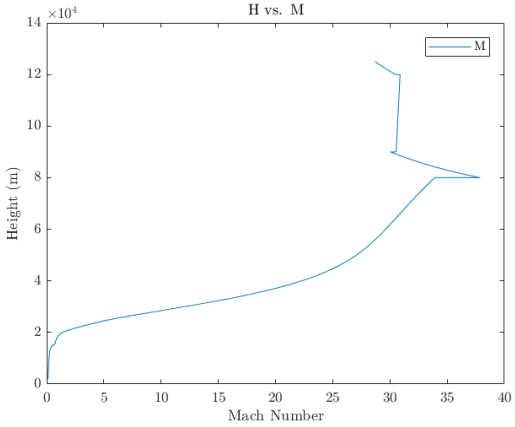
(d) $\dot{\theta}$ without damping

Figure 4.9: θ and $\dot{\theta}$ evolution vs. time

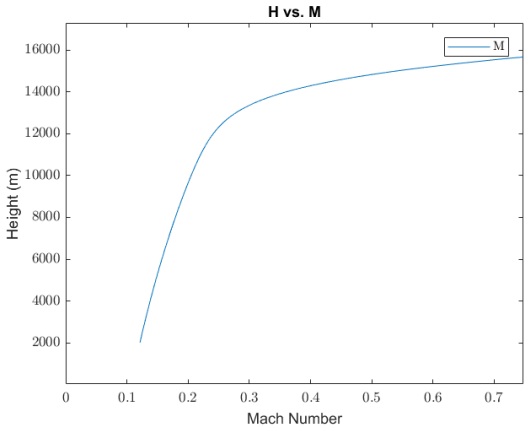
As expected, the both parameters present spurious oscillations without damping.

4.1.3 Complete descent results

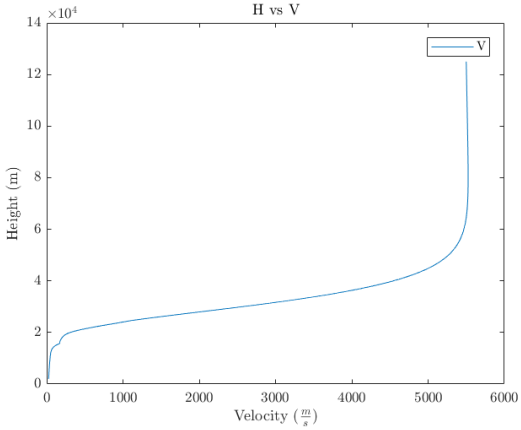
Finally, the results for the whole EDL phase are presented for the selected geometry, which will be an aeroshell of **45 ° and 3.5 m**, and a parachute with a diameter of **8 m**. The aerodynamics are axisymmetric. They can be consulted at Figure 4.10:



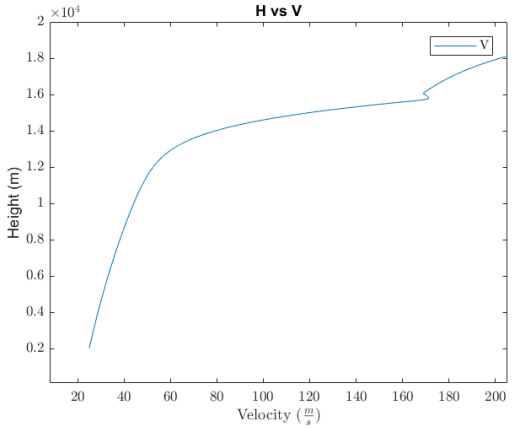
(a) H vs. M complete descent



(b) H vs. M detail during parachute's descent



(c) H vs. V complete descent



(d) H vs. V detail during parachute's descent

Figure 4.10: Complete decent for final configuration

Conclusions

The aim of this thesis was to calculate the entry, descent and landing phases of a small mass capsule in the martian atmosphere. The propposed solution included the development of a 3 DoF flight simulator. The validation of such simulator using analytical trajectories was successful. Moreover, the aerodynamic data was calculated using a nummerical method based on an Eulerian approach. In order to validate both the scheme and the mesh, an experimental case has been modelled under the same conditions, and the obtained results proved satisfactory the propposed nummerical solution. The CFD cases have been solved using OpenFOAM.

Regarding the flight simulator, the propposed 3 DoF model has proven to be succesful. The Newtonian model, as well as the treatment of the variables and the reference frame selected to solve the equation has been adequate. Thanks to the validation case, a wide range of timesteps could be chosen between two nummerical schemes, selecting a balanced option between accuracy and computational cost. It is important to note that, after the consideration of a 2D case and an axisymmetric one, the effect of the lift is important. However, it would be necessary to carry out a 3D simulation (which was outside the scope of this study) in order to compute the real effect of the coefficients (in Appendinx F it can be oberved how 2D coefficients are larger than axisymmetric ones).

After finishing this section, the milestone of the project was reached: the aerodynamic design. It presented both technical difficulties and a high workload. Several issues were encountered during the process, specially in the validation of the mesh and the solver and during the development of the transonic analysis. Despite this problems, the final results have been very satisfactory, as they behaved in a similar way like other thesis and they provided a final aerodynamic model which performed well in the final simulations. Regarding the parachute design, it followed experimental data and due to the high amount of missions using the type of parachutes, the development has also been successful.

Finally, regarding the final simulations, during the first iterations severe oscillations were observed, which made necessary to include a damping coefficient in each stage of the flight. These coefficients have been determined experimentally. These coefficients

have been determined experimentally. The results showed that, using 2D coefficients and the propposed parachute from (Pasolini et al., 2017), the landing velocity was achieved $10000m$ above the martian surface. This is probably due to the 2D assumption, which makes the shock waves to be stronger, and so are the coefficients (at $\alpha = 0$, the drag coefficient in the 2D assumption is a 30% larger than in the axisymmetric case). This is the reason that have led to use the axisymmetric data and a smaller parachute, where a reasonable landing altitude have been obtained (approximately $2000m$, where the Viking and the Pathfinder missions landed (Braun, 2010)).

4.1 Future work

Several features of this project could be expanded in future studies. First of all, the aerodynamic model is calculated using the assumption that the martian atmosphere is calmed. Therefore, a atmospheric model which included wind effects will be of high interest, in order to improve the trajectory. Finally, if such supposition was made, a 6 DoF flight simulator should be necessary, as well as a 3D computational analysis.

Regarding the computational model, one such important factor is the viscosity of the fluid. In an inviscid approach, no turbulence model is introduced, which makes very difficult to analyse certain aspects of the flow behaviour, such as the wake. If this project reached further stages, it should be interesting to introduce dissipative effects because the aerodynamic coefficients could be studied easier. Moreover, the geometry of the aeroshell inside the wake could be improved, which is important specially for stability issues. Another important point that should be furtherly studied is the transonic model, which is known to be very complex and requires of much more time in order study it properly (for example, the oscillatory behaviour of the wake). Finally, regarding the effect of the temperature, a study on the materials to construct the capsule should be carried on.

Regarding the simulator results, a 3D analysis taking into account the lift of the capsule shuld be done, in order to compute the real effect of this force (which in a 2D approximation has shown to be relevant) and in order design a proper parachute, which diameter should change when 3D shock-wave effects and viscous ones are taken into account.

And finally, the last item that should be furtherly studied is the damping of both phases. In this thesis, a basic damping has been introduced. For this type of descents, it would be necessary to design this aeroshell's feature using more specific methods, or even considering active stability systems.

Planification

One important aspect of a thesis is the planification. One detailed planification plan has been presented in the Project Charter. In Figure 4.11, the final Gantt Chart is presented, with all the tasks updated accordingly with the final items developed:

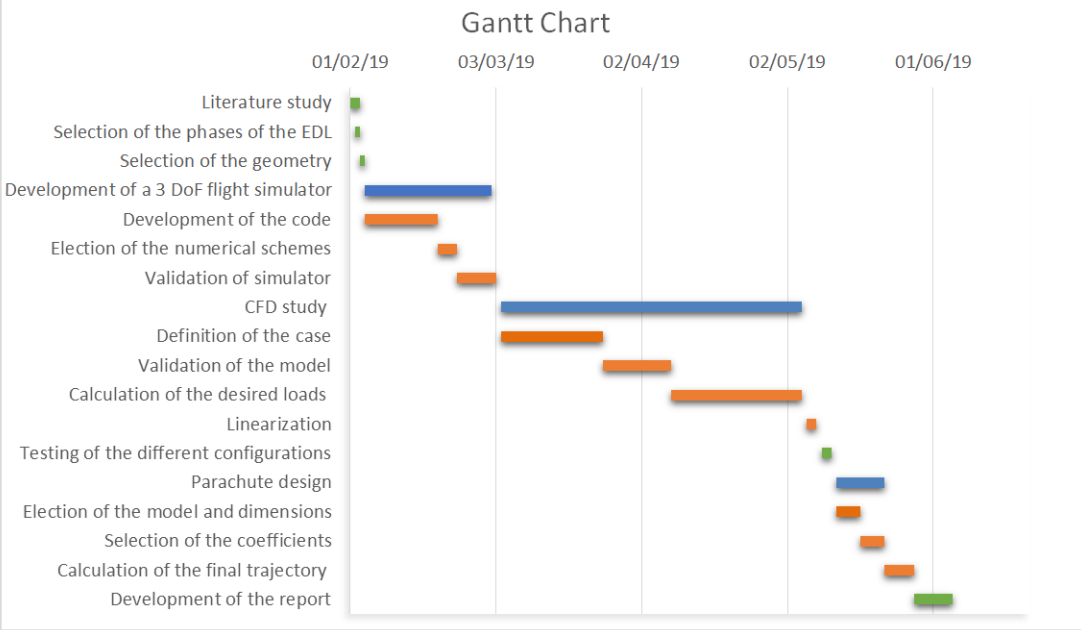


Figure 4.11: Gantt chart

As it can be seen, most of the time has been spent in CFD analysis, because of its complexity and computational time. The other most time demanding section has been the flight simulator programming, as the validation required several iterations until its completion.

Enviromental Impact

Regarding the spatial exploration field, a lot of debris is accumulated on its surface, mainly due to the difficulty of removing it. This thesis studied several configurations, and it finally choosed the best solution amongst smaller dimensions and higher performance. With this criteria, the amount of debris that this project would accumulate ofer the martian surface would be minimal.

Finally, another important aspect of this design is that the descent and the stability is calculated without additional control systems. This fact would reduce the amount of fuel used inside the atmosphere (in later stages, however, it would be likely to incorporate such systems, but at least an option is presented without them).

Budget

In order to compute the cost of this project, a table is presented. The average hour price has been set to 20. It has also been taken into account the cost of the licenses of the software used and the electricity consumed for the computer due to the simulations. The working hours total price take into account a 30% corresponding to taxes such us insurances. The total cost of this project is shown in Table 4.2:

HUMAN RESOURCES			
Position	**€ / hour**	**Total hours**	**Total (€)**
Engineer	20	800	20800
SOFTWARE LICENSES			
Software	**Hours used**	**Type of license**	**Total (€)**
OpenFOAM	1000	open-source	0
Matlab	400	student version	0
OTHER RESOURCES			
Product	**KW/H per day**	**Cost per day (€)**	**Total (€)**
Laptop power (150 days used)	4	0.145	90
TOTAL COST OF THE PROJECT			
20890 €			

Table 4.2: Budget of the project

It is important to note that, in order to calculate the energy consumed by the computer, it has been considered to waste 1.400 kw/H per year (the average for a gaming computer), due to the high usage it has been required, specially from the simulations.

Bibliography

Ames. Tables for compressible flow. 1951.

John D. Anderson. Computational fluid dynamics, 1995.

Michel Van Bizen. Differential Equation - 1st Order Solutions (8 of 8) How to Calculate Parachutist's Terminal Speed, 2015. URL https://www.youtube.com/watch?v=DS78wFkrBmE{&}list=PLd1FdIMX1H0FtwxA7ktwaj54jeQEmQTXP{&}index=2{&}t=0s.

Alexander E Bondarev. Analysis of the Accuracy of OpenFOAM Solvers for the Problem of Supersonic Flow Around a Cone. pages 221–230, 2018.

Robert D Braun. Mars Exploration Entry , Descent and Landing Challenges. pages 1–32, 2010.

Jordi Casacuberta. OpenFOAM GUIDE FOR Begginers. 2014.

Wang Chao, Liu Xinyu, and Li Feng. Six-DOF Modeling and Simulation for Generic Hypersonic Vehicle in Reentry Phase. *Procedia Engineering*, 99:600–606, 2015. ISSN 1877-7058. doi: 10.1016/j.proeng.2014.12.577. URL http://dx.doi.org/10.1016/j.proeng.2014.12.577.

D J Cockrel, Leicester Le, and A D Young. *The Aerodynamics of Parachutes*. Number 295. 1987. ISBN 9283504224.

NASA Company. Nasa Mars Atmospheric Model. URL https://mars.jpl.nasa.gov/MPF/science/weather.html.

C F D Consulting. Boundary Conditions in OpenFOAM. (May), 2017.

Davis Crawford. Investigation of the laminar aerodynamic heat-transfer characteristics of a hemisphere-cylinder in the langley 11-inch hypersonic tunnel, 1956.

Juan R Cruz, J Stephen Lingard, Aerospace Engineer, Hampden House, Monument Business Park, and United Kingdom. Aerodynamic Decelerators for Planetary Exploration :. pages 1–20, 2018.

CTTC. Numerical resolution of the generic convection-diffusion equation. pages 1–28, 2019.

Karl T Edquist, Prasun N Desai, and Mark Schoenenberger. Aerodynamics for the Mars Phoenix Entry Capsule. pages 1–18, 2019.

Scott G V Frendreis. AIAA 2009-5601 Six-Degree-of-Freedom Simulation of Hypersonic Vehicles Three-Dimensional Vehicle Simulation Framework. (August), 2009.

Christopher J Greenshields, Henry G Weller, Luca Gasparini, and Jason M Reese. Implementation of semi-discrete , non-staggered central schemes in a colocated , polyhedral , finite volume framework , for high-speed viscous flows. 2009. doi: 10.1002/fld.

R I Issa. Solution of the Implicitly Discretised Fluid Flow Equations by Operator-Splitting. 65:40–65, 1981.

Theo W Knacke. *Parachute Recovery sytems design manual*. 1991. ISBN 0915516853.

Alexander Kurganov and Eitan Tadmor. New High-Resolution Central Schemes for Nonlinear Conservation Laws and Convection – Diffusion Equations. 282:241–282, 2000. doi: 10.1006/jcph.2000.6459.

J Stephen Lingard. Ram-Air Parachute Design. (May):1–51, 1995.

S Lingard. A semi-empirical theory to predict the load-time history of an inflating parachute. 1984. doi: 10.2514/6.1984-814.

William P Ludtke. A technique for the calculation of the opening-shock forces for several types of solid cloth parachutes. 1972.

Marcantoni, Alberto Cardona, Paul H Kohan, Ricardo D Quinteros, and Mario A Storti Eds. High speed flow simulation using openfoam. XXXI:13–16, 2012.

R C Mehta. Numerical simulation of supersonic flow past reentry capsules. 15:31–41, 2006. doi: 10.1007/s00193-005-0003-0.

C.L Merkel and P E Buelow. The relationship between pressure-based and density-based algorithms. 1992.

Moog. Qualification flight tests of the Viking. (73):0–13, 1973.

Cathleen Synge Morawetz. Mixed equations and transonic flow. 1(1):1–26, 2004.

Gino Moretti, P O Box, and New York. Computation of Compressible Flows. (Lax 1954), 1987.

company NASA. NASA Mars Exploration Overview. URL https://mars.nasa.gov/programmissions/overview/.

Ben E Nikaido. OpenFOAM Simulations of Atmospheric-Entry Capsules in. (January): 1–14, 2015.

OpenFOAM. OpenFoam Website. URL https://www.openfoam.com/.

P. Pasolini, F. Punzo, F. Paudice, D. de la Torre, N. Cimminiello, C. Molfese, R. Savino, I. Roma, G. Zuppardi, M. Grassi, P. Dell'Aversana, L. Gramiccia, R. Aurigemma, E. Fantino, F. Causa, and F. Esposito. The Small Mars System. *Acta Astronautica*, 137(April 2017):168–181, 2017. ISSN 00945765. doi: 10.1016/j.actaastro.2017.04.024. URL http://dx.doi.org/10.1016/j.actaastro.2017.04.024.

TJ Poinsot and SK Lele. Boundary Conditions for Direct Simulations Compressible Viscous Flows. pages 104–129, 1992.

Shiva Prasad and G Srinivas. Flow Simulation over Re-Entry Bodies at Supersonic & Hypersonic Speeds Flow Simulation over Re-Entry Bodies at Supersonic & Hypersonic Speeds. (July 2012), 2014.

Mark Schoenenberger and Eric M Queen. Limit Cycle Analysis Applied to the Oscillations of Decelerating Blunt-Body Entry Vehicles. 2019.

LF Shampine. Stability Of Runge Kutta Methd. 1984.

J Sinclair, X Cui, J Sinclair, and X Cui. A theoretical approximation of the shock standoff distance for supersonic flows around a circular cylinder A theoretical approximation of the shock standoff distance for supersonic flows around a circular cylinder. 026102, 2017. doi: 10.1063/1.4975983.

Nathan Slegers, Nathan Slegers, and Mark Costello. Aspects of Control for a Parafoil and Payload System Aspects of Control for a Parafoil and Payload System. 2003.

George Solt and Alexan Ria. Performance of and Design Criteria for Deployable Aerodynamic Decelerators. 1961.

M Sun and K Takayama. Vorticity production in shock difraction. 2002. doi: 10.1017/S0022112002003403.

J Tannehill and A Anderson. Computational Fluid Mechanics, 1997.

Miguel Ángel Gómez Tierno. *Mecanica del vuelo*. 2012.

L L Trimmer. Study of the blunt body stagnation point velocity gradient in hypersonic flow. (May), 1968.

H K Versteeg and Malalasekera W. *An Introduction to Computational Fluid Dynamics*. 2007. ISBN 9780131274983.

Antonio Viviani, Giuseppe Pezzella, Carmine Golia, and Seconda Università. Aerothermodynamic field past a reentry capsule for sample return missions. pages 1–13, 2012.

Zenghui Zhang, Lingyu Yang, Youwu Zhong, and Gongzhang Shen. Modeling and Analysis for a Generic Hypersonic Vehicle *. (60804007):152–158, 2010.

Polythecnical University of Catalonia (UPC)

Escola d'Enginyeria Industrial, Aeroespacial i Audiovisual de Terrassa (ESEIAAT)

Appendixes

A Thesis submitted by Oscar Toledo Farrando for the degree of Aerospace Engineer in the UPC.

Supervised by:
Enrique Ortega

A Description of the algorithms

The solvers OpenFOAM uses are based in well-known algorithms, these are, PISO, SIMPLE and PIMPLE. Moreover, as the selected scheme is a density based on (OpenFOAM), the difference between it and a pressure based one will also be presented.

A.1 The PISO algorithm

It stands for Pressure Implicit with Splitting of Operators. This is a non-iterative method for solving fluid flow equations which are implicitly discretised and time dependent. This method is on the use of pressure and velocity as dependent variables, so it can be both applied to compressible and incompressible forms of the transport equations. Therefore, it is a pressure based solver.

The main characteristic of this scheme is the splitting of the final solutions, where in each step operations involving the pressure are decoupled from the ones involving the velocity. The accuracy depends on the number of splits used, and the errors in the solution decay rapidly if such procedure is refined (Issa, 1981).

Moreover, this method is also fairly stable for large time-steps, which makes it perfect for steady-state solutions. A general flowchart for this method is given in 4.12:

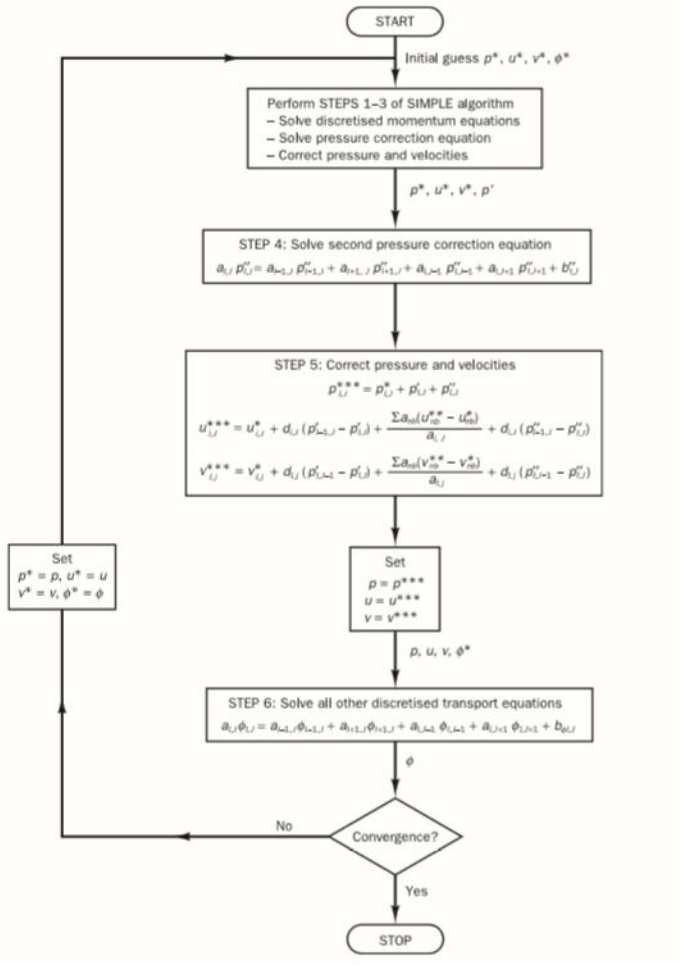


Figure 4.12: PISO flowchart

A.2 The SIMPLE algorithm

It stands for Semi-Implicit Method for Pressure-Linked Equations (Versteeg and W, 2007). This is an iterative method for the calculation of pressure. In order to calculate the velocity, the SIMPLE algorithm uses a guessed pressure $p*$ and iterates until calculating the velocity components.

This iterative methods needs to correct the pressure on every loop, which make it susceptible to divergence unlesss an under-relaxation factor is introduced. As usually the momentum equations need to be solved sequentially, this is also a pressure-based method.

A general flowchart is presented in 4.13:

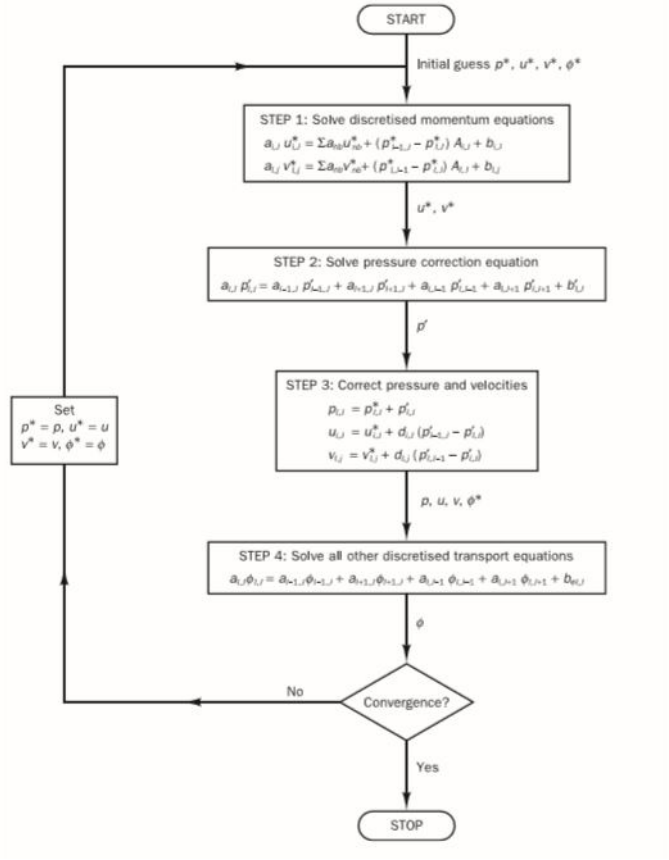


Figure 4.13: SIMPLE flowchart

This algorithm has had multiple reviews, amongst where it can be highlighted the following ones:

- SIMPLER (SIMPLE Revised): this improved version substitutes the pressure correction for a discretised equation for pressure. This makes the obtaining of the pressure faster.
- SIMPLEC (SIMPLE Consistent): in this case, the momentum equations are manipulated and therefore, the velocity correction equations omit less significant elements.

A.3 Central upwind schemes (CUS)

Based on the method proposed by (Kurganov and Tadmor, 2000), is the scheme over which the *rhoCentralFoam* selected scheme is based.

These schemes offer universal finite-difference methods for solving the hyperbolic convection-diffussion equations. As they are not linked in the eigenstructure of the problem, they

can be implemented to solve large gradient phenomena such as shock waves.

Usually these schemes have high nummerical viscosity, but the selected solver is based on a a modified version implemented by (Kurganov and Tadmor, 2000) which reduces this fact.

A.4 Comparision between pressure-based solvers and density based solvers

One one hand, there are the pressure-based methods. They were formulated for incompressible flows at low Reynolds numbers, although they have been extended to compressible applications (Merkel and Buelow, 1992).

They employ an algorithm which belongs to the projection method. It is based in achieving the continuity of the velocity field by solving a pressure equation. This equation is derived from the momentum and the continuity equation in a manner that the obtained velocity field accomplishes the conitnuity (Merkel and Buelow, 1992).

Here it is presented the segregated algorithm, whihc solves the non-linear equations sequentially. The flowchart can be observed in 4.14:

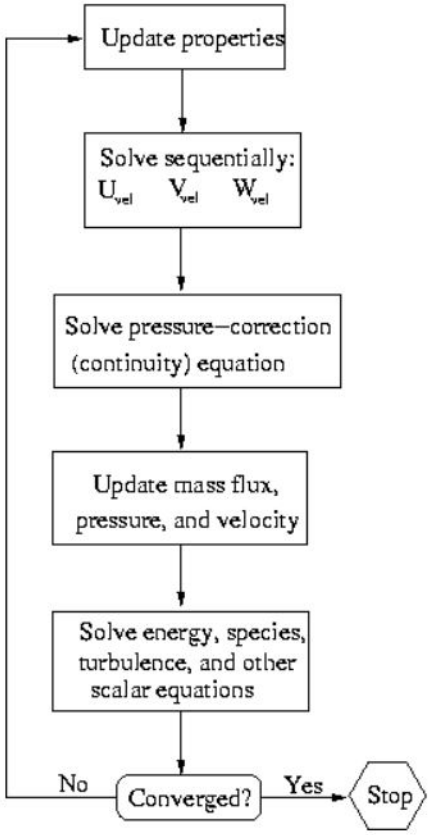


Figure 4.14: Pressure based method flowchart

On the other hand, schemes such as *rhoCentralFoam* use a density-based method. This methods, initially developed for external, transonic aerodynamics, were developed for inviscid, compressible flows, although they have been extended to viscous ones. The implicit formulations of this scheme solve the momentum, continuity and energy equations in a simultaneous manner (Merkel and Buelow, 1992).

In this method, the non-linear governing equations are discretised in each cell. They are solved in each time step to update the solution. Finally, as it uses an implicit scheme, the equations are solved at the same time in order to obtain all the solutions.

The general algorithm this method follows is presented below:

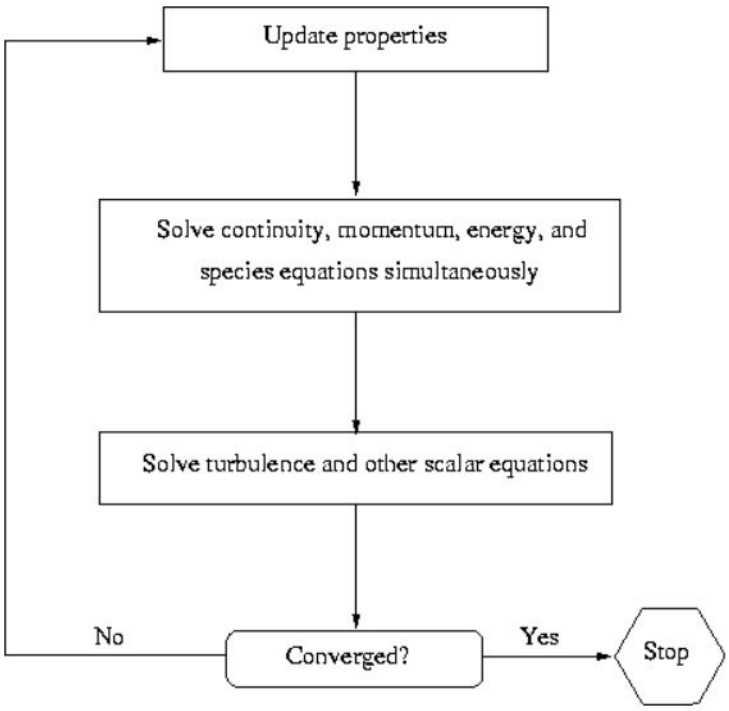


Figure 4.15: Density based method flowchart

B Standoff distance of shock waves

The distance between the shokwave and a blunt body is inversely proportional to the Mach number. This effect, which may seem paradoxic, has a physical explanation that agrees with the well known **Newton law**.

According to (Sinclair et al., 2017), the detachment distance, known as δ_{sd}, depends on various factors, i.e, the Mach number, the body profile and gas properties. While the best way to predict this standoff shock distance is the experimental one, an analytical law will be presented as a base to estimate δ_{sd}: the modified Newtonian law (the whole study will be done using a cylindrical geometry, which will be supposed to behave as a blunt body like the one presented in this thesis, in Figure 4.16).

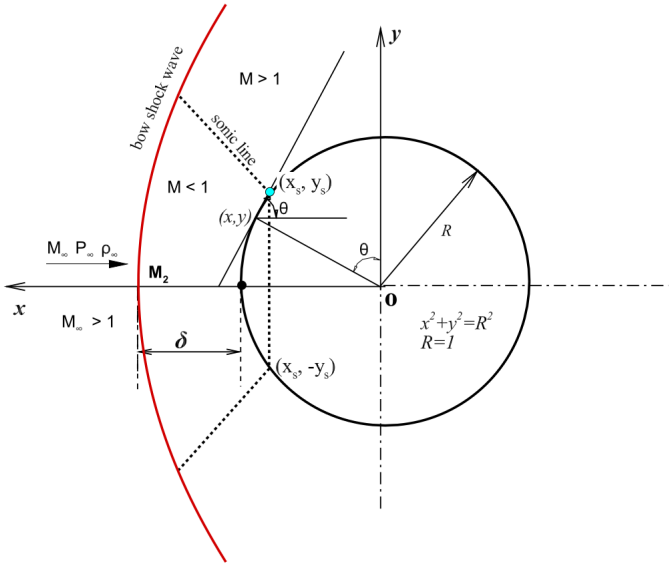


Figure 4.16: Geometry and flowfield parameters (source: (Sinclair et al., 2017))

B.1 Modified Newton Law

First of all, and in a similar way like in (Trimmer, 1968), the Newtonian impact theory can be presented for blunt body as follows:

$$\frac{C_p}{C_{p,max}} = \sin\theta^2 \tag{71}$$

where, in this case, θ is the angle between the flow direction and the tangent with the surface. The Cp can be expressed as

$$Cp = \frac{2}{\gamma_{heat} M_\infty^2}\left(\frac{p}{p_\infty} - 1\right) \tag{72}$$

and denoting p_2 the pressure conditions after the shock, the normal shock relation can be stated as:

$$\frac{p_2}{p_\infty} = 1 + \frac{2\gamma_{heat}}{\gamma_{heat}+1}(M_\infty^2 - 1) \tag{73}$$

As the flow after the shock can be considered isentropic, the stagnation pressure p_{02} is constant between the forebody and the shock. Thus:

$$\frac{p_{02}}{p_2} = \left(1 + \frac{\gamma_{heat}-1}{2} M_2^2\right)^{\frac{\gamma_{heat}}{\gamma_{heat}-1}} \tag{74}$$

and the Mach number in the normal-shock:

$$M_2 = \sqrt{\frac{2 + (\gamma_{heat} - 1)M_\infty^2}{2\gamma_{heat}M_\infty^2 - \gamma_{heat} + 1}} \tag{75}$$

If the Mach number is known in a surface point, then it can be stated that

$$\frac{p_{02}}{p} = \left(1 + \frac{\gamma_{heat} - 1}{2}M^2\right)^{\frac{\gamma_{heat}}{\gamma_{heat}-1}} \tag{76}$$

With Equations 73, 74 and 76 the static C_p can be obtained if

$$\frac{p}{p_\infty} = \frac{p}{p_{02}}\frac{p_{02}}{p_2}\frac{p_2}{p_\infty} \tag{77}$$

Therefore, the stagnation pressure coefficient, $C_{p,max}$, achieved at $M = 0$ is

$$\begin{aligned} C_{p,max} = &\frac{2}{\gamma_{heat}M_\infty^2}\left(\left(\frac{(\gamma_{heat}+1)^2M_\infty^2}{4\gamma_{heat}M_\infty^2 - 2(\gamma_{heat}-1)}\right)^{\frac{\gamma_{heat}}{\gamma_{heat}-1}}\right. \\ &\left.\left(\frac{1 - \gamma_{heat} + 2\gamma_{heat}M_\infty^2}{\gamma_{heat}+1}\right) - 1\right) \end{aligned} \tag{78}$$

In a similar manner, at the sonic point where $M_s = 1$, the corresponding pressure coefficient C_{ps}, introducing Equations 73 and 74, is

$$\begin{aligned} C_{ps} = &\frac{2}{\gamma_{heat}M_\infty^2}\left(\left(\frac{\gamma_{heat}+1}{2}\right)^{-\frac{\gamma_{heat}}{\gamma_{heat}-1}}\left(\frac{(\gamma_{heat}+1)^2M_\infty^2}{4\gamma_{heat}M_\infty^2 - 2(\gamma_{heat}-1)}\right)^{\frac{\gamma_{heat}}{\gamma_{heat}-1}}\right. \\ &\left.\left(\frac{1 - \gamma_{heat} + 2\gamma_{heat}M_\infty^2}{\gamma_{heat}+1}\right) - 1\right) \end{aligned} \tag{79}$$

where it has been also introduced the pressure relation at the sonic point

$$\frac{p_s}{p_{02}} = \left(\frac{\gamma_{heat}+1}{2}\right)^{-\frac{\gamma_{heat}}{\gamma_{heat}-1}} \tag{80}$$

B.2 Approximation of the shock standoff distance

B.2.1 Sonic tube's cross section function

As shown in (Sinclair et al., 2017), the standoff shock distance δ_{sd} is related with the ratio of free stream density and the density after the shock as

$$\frac{\delta_{sd}}{D} = 0.41\frac{\rho_\infty}{\rho_2} \tag{81}$$

This equation assumes a linear density profile, which agree with the simulation results of (Sinclair et al., 2017) at high Mach numbers, but fails to predict low Mach numbers (where the distance tends to go to infinite, but Equation 81 is limited to 0.41 at $M = 1$).

In order to correct that, a different approach is used: a "sonic-tube" is considered (see Figure 4.18), where the conservation of mass is accomplished and can be written as

$$\dot{m}_3 = \rho_3 A_3 V_3 = \frac{p_{03} A_3 M_3 \sqrt{\gamma_{heat}}}{\sqrt{R_g T_{03}}} \left(1 + \frac{\gamma_{heat} - 1}{2} M_3^2\right)^{-\frac{\gamma_{heat}+1}{2(\gamma_{heat}-1)}} = constant \tag{82}$$

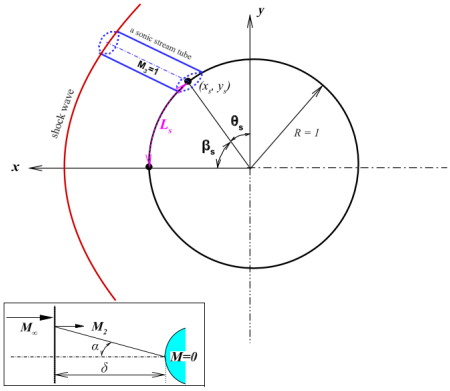


Figure 4.17: Schematic of the "sonic tube" approach (source: (Sinclair et al., 2017))

where it has been introduced the ideal gas state equations and the normal shock relations for pressure and temperature like Equation 74.

As the after-shock subsonic region depends on M_∞, therefore the subsonic region can be considered as

$$b = \frac{A_3}{\theta} = \frac{A_3}{\arccos y_s} \tag{83}$$

b is infinitely large for $M_\infty \longrightarrow 1$ to a limited value when $M_\infty \longrightarrow \infty$. Therefore, the rate of reduction of this area can be expressed as

$$b_{ys} = \frac{db}{dy_s} = \frac{A_3}{\sqrt{1 - y_s^2}(\arccos y_s)^2} \tag{84}$$

and introducing $x_s = \sqrt{1 - y_s^2} = \cos\beta_s$ and $y_s = \cos\theta_s$ from Figure 4.18

$$b_{ys} = \frac{A_3}{\theta^2 \cos\beta_s} \tag{85}$$

and as A_3 can be related to the arc longitude from the stagnation point to the sonic region (which in this case, is an arc of a cylinder of radius 1), it can be stated that $A_3 = \beta_s^2$ (Sinclair et al., 2017), yielding:

$$b_{ys} = \frac{\beta_s^2}{\theta_s^2 \cos \beta_s} \tag{86}$$

Equation 86 is promising because it maintains a constant value for high Mach numbers and tends to infinite when the Mach number tends to 1.

B.2.2 Standoff shock distance function

Following Figure 4.18, the standoff distance δ_{sd} can be written as

$$\delta_{sd} = \frac{M_2}{\tan \alpha} \tag{87}$$

In order to easily compute the dependence of Equation 87, a range of α can be defined as $\alpha \in [0, \frac{\pi}{2}]$. Therefore, $\tan \alpha$ varies from 0 for $M_\infty \longrightarrow 1$ to ∞ for $M_\infty \longrightarrow \infty$ (Sinclair et al., 2017).

In a similar manner as for the cross section rate change b_{ys}, a reduction parameter is introduced for δ_{sd} in Equation 88:

$$\delta_{sdM2} = \frac{\partial \delta}{\partial M_2} = \cot \alpha \tag{88}$$

This equation shows the behaviour of the standoff distance with respect to the Mach number. When $M_\infty \longrightarrow 1$, the parameter α is close to 0 and therefore δ_{sdM2} tends to ∞. Moreover, when M_∞ turns hypersonic, α gets close to $\frac{\pi}{2}$, making $\cot \alpha \longrightarrow 0$ and therefore stabilizing δ_{sd} to a constant value.

Acording to (Sinclair et al., 2017), a good way to "control" the variation of δ_{sd} is to relate the standoff shock distance with the cross section of th subsonic distance after a shock. For doing so, they propose

$$\delta_{sdM_2} = b_{ys} \longrightarrow \cot \alpha = \frac{\beta_s^2}{\theta_s^2 \cos \beta_s} \tag{89}$$

Finally, using Equation 87

$$\delta_{sd} = \frac{M_2 \beta_s^2}{\theta_s^2 \cos \beta_s} \tag{90}$$

And introducing Equation 75 into Equation 90

$$\delta_{sd} = \frac{\beta_s^2}{\theta_s^2 \cos\beta_s}\sqrt{\frac{2+(\gamma_{heat}-1)M_\infty^2}{2\gamma_{heat}M_\infty^2-\gamma_{heat}+1}} \tag{91}$$

where δ_{sd} is already a non-dimensionalized equation. Figure **??** shows how this function fits the experimental results in a excellent way (in the mentioned figure, the Equation 91 is represented by the black continuous line).

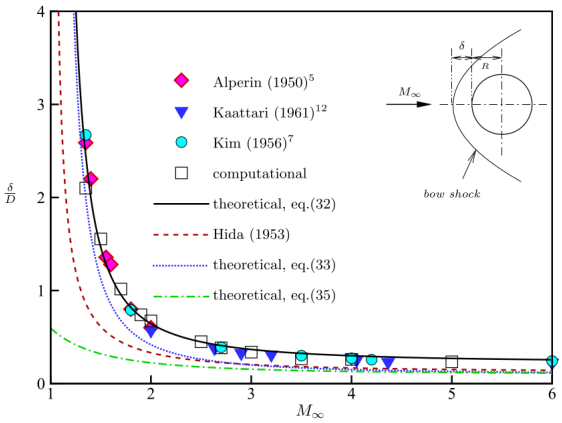


Figure 4.18: Comparision of existent experimental data with Eq 91 (source: (Sinclair et al., 2017))

C Vorticity in inviscid flow at the wake of blunt bodies

The appearence of vorticity in compressible flow is mainly due to shock difractions. This section will present the procedure followed by (Sun and Takayama, 2002), where several analysis are carried out to characterise the behaviour of vorticity in an inviscid flow around sharp corners.

The geometry that is going to be tested is shown in Figure 4.19. The angle θ is the corner angle and will be changed in order to study its effect in vorticity appearence.

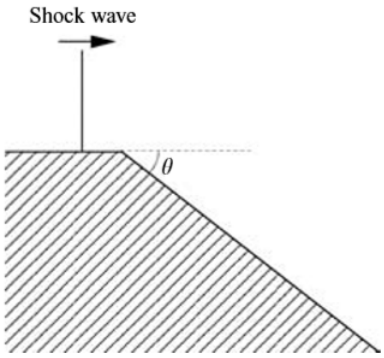


Figure 4.19: Schema of an expansion over a conrner (source: (Sun and Takayama, 2002))

An Eulerian approach is followed. Its complete development is shown in (Sun and Takayama, 2002).

C.1 Characterisation of vorticity

The strength of a vortex can be expressed by the circulation Γ as

$$\Gamma = \int_s \omega ds \tag{92}$$

The integral can be also expressed as

$$\Gamma = \int_L V dL \tag{93}$$

This is important because in the study section there are velocity discontinuities and the Green theorem could not be properly applied in Equation 92.

When simulations are carried out, it is observed that circulation increases linearly during time, except at its origin (Sun and Takayama, 2002). This is due to the equations used. In an Eulerian scheme, the diffraction is self-similar, which means that physical variables of pressure, density and velocity are functions of $(\frac{x}{t}, \frac{y}{t})$. Therefore, the circulation can be expressed as a ratio of time. It will depend on the incident Mach number M_s and the difraction angle (if the gas properties are given):

$$\frac{\Gamma}{t} = f(M_s, \theta) \tag{94}$$

This ratio appeared to be unchanged with different schemes (central or upwind) and different mesh types (triangular or quadrilateral).

C.2 Theoretical basis

According to (Sun and Takayama, 2002), the vortex in inviscid flows are produced by the roll-up of the flow near the wall (or slipstream) that initiates in the corner due to shock diffraction. As the circulation must be conserved, the rate of vorticity production should be equal to the vorticity entering in the flow. This circulation ration, therfore, can be expressed as a difference between the two tangential velocities on each side of the slipstream (inside and outside the wake):

$$\frac{\Gamma}{t} = \int_{\delta} u \frac{du}{dy} dy = \frac{1}{2}(U^2_{\delta=\delta_{max}} - U^2_{\delta=0}) \tag{95}$$

In Equation 95, δ is the assumed thickness of the slipstream. When the diffraction occurs, the flow close to the corner has a velocity almost equal to 0, therefore $U_{\delta=0} = 0$, yielding

$$\frac{\Gamma}{t} = \frac{1}{2} U^2_{\delta=\delta_{max}} \tag{96}$$

The velocity U_1 behaves as shown in Figure 4.20. Regions 1 and 2 correspond to the fow in front of and behind of the incident shock produced to diffraction. The pressure in region 2 is higher than in region 3, where the flow is accelerated due to expansion waves.

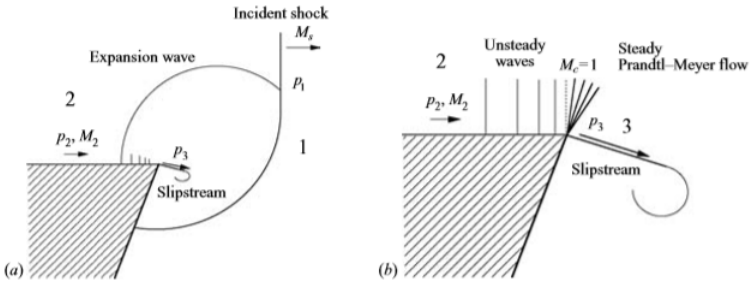


Figure 4.20: Wave structure in weak shock wave diffraction (a), general view; b) closeup in the corner (source: (Sun and Takayama, 2002))

The velocity U_1 can be calculated following the model presented in (Sun and Takayama, 2002), but is not presented here because the aim of this section is to describe qualitatively the behaviour of vorticity after sharp corners in inviscid flow.

C.3 The effect of θ

From experimental results presented in Figure 4.21, it can be seen that the vorticity produced gows rapidly from 30° to 60° but, when the wall angle is greater than 90°, the vortex shape remains almost constant (this comparative effect is plotted in Figure 4.22). According to (Sun and Takayama, 2002), this is because a large wall angle allows

a large creation of vorticity.

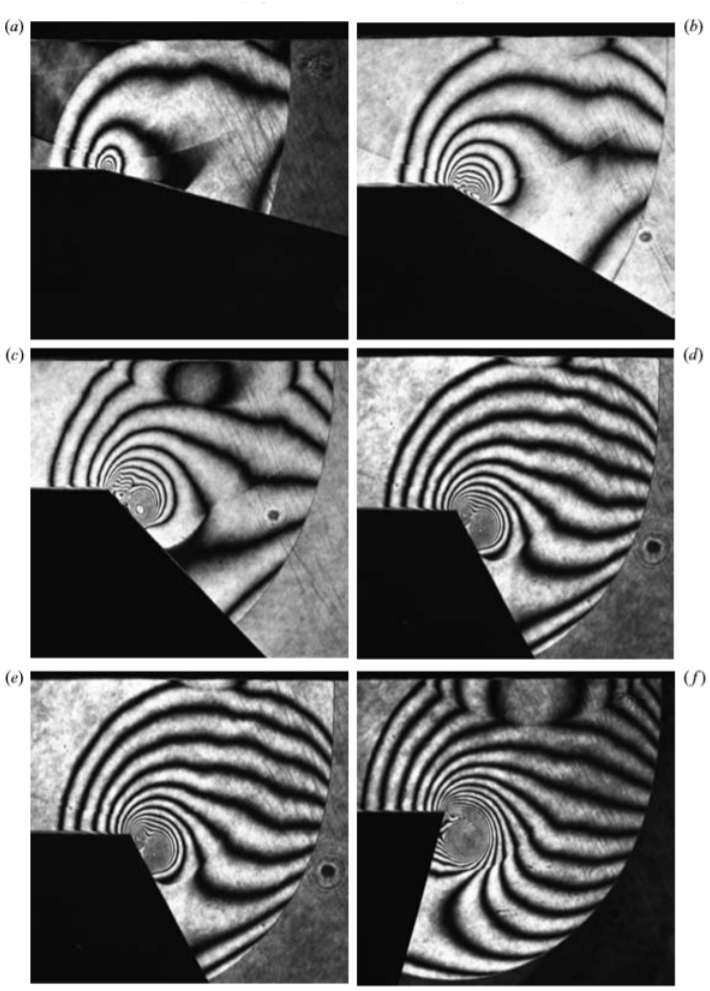


Figure 4.21: Results of shock diffraction: a) $\theta = 15°$, M = 1.51; b) $\theta = 30°$, M = 1.5; c) $\theta = 45°$, M = 1.5; d) $\theta = 60°$, M = 1.4; e) $\theta = 90°$, M = 1.4; f) $\theta = 105°$, M = 1.4; (source: (Sun and Takayama, 2002))

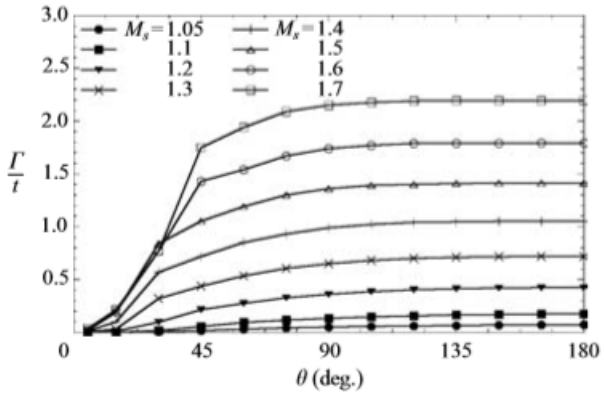


Figure 4.22: Comparision of vorticity rates at different corners shapes and Mach number (source: (Sun and Takayama, 2002))

Moreover, the nummerical results presented in Figure 4.23 show a good accuracy with experimental results, which assures the use of Euler equations to reproduce the diffracted shock, the expansion waves and the shape of the vortex.

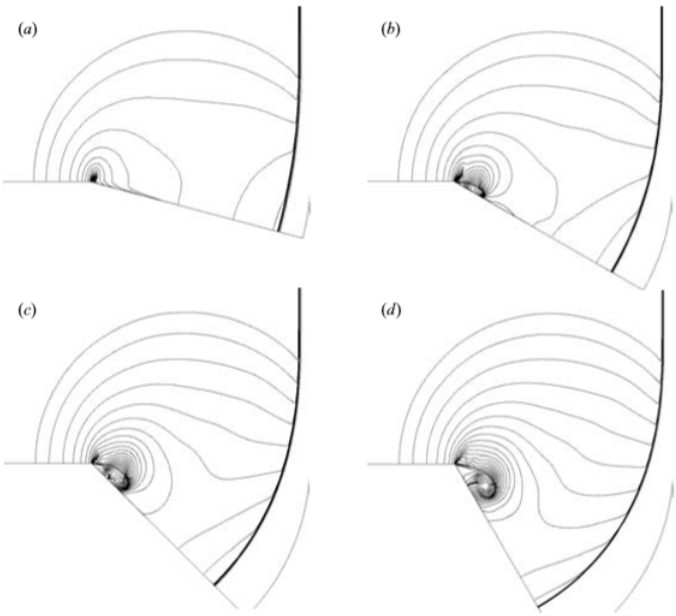


Figure 4.23: Density contours at M = 1.5: a) $\theta = 15°$, b) $\theta = 30°$, c) $\theta = 45°$, a) $\theta = 60°$ (source: (Sun and Takayama, 2002))

Finally, one important fact to take into account is that in Figures 4.21 d) and 4.21 f) a secondary vortex appears, and in the corresponding nummerical ones it does not. This is because the primary vortex is much stronger, and an Eulerian approach is not

capable of detecting it, because a viscous Navier-Stokes analysis would be needed (Sun and Takayama, 2002).

C.4 Incident shock effect

The last parameter affecting the circulation ratio is the incident shock strength. In order to calculate that, a 135° corner is simulated at different Mach numbers. Figure 4.24 plotts the the shape of the vortexs at Mach numbers from 1.23 to 3. According to (Sun and Takayama, 2002), for low Mach numbers the vortex are well defined (Figure 4.24a-b), but at higher Mach numbers, the secondary vortex previously mentioned becomes stronger, thus penetrating in the primary one and distorting it (Figure 4.24a-b).

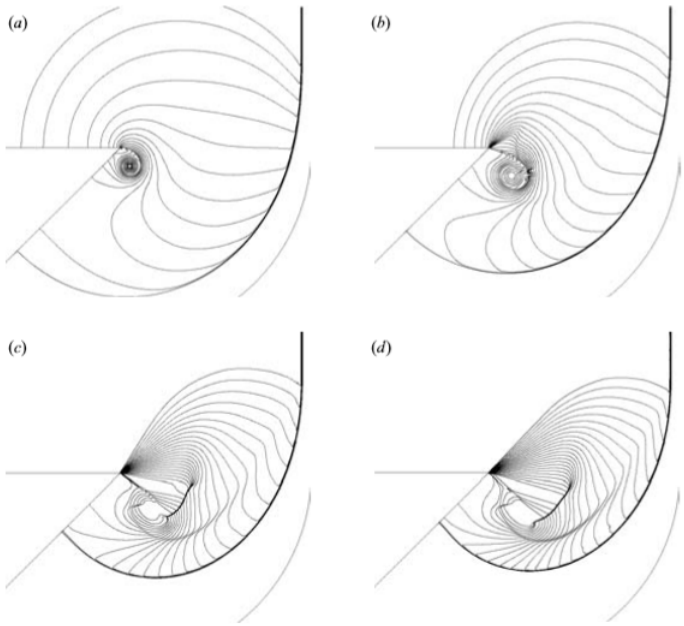


Figure 4.24: Shock diffraction at $\theta = 135 : a)M = 1.23, b)M = 1.6, c)M = 2.43, a)M = 3(source : (SunandTakayama, 2002))$

Finally, Figure 4.25 shows the effect of incident shocks in the rate of circulation. As well as in the wall angle, it the rate increases strongly up to 45°. It can be also observed that from 45° to 180°, the vorticity grows monotonically, however, its is observed that at 30°, there is a peak in $M = 1.5$ and after that the circulation rate decreases. According to (Sun and Takayama, 2002), this is due to dissipative effects of the secondary vortex.

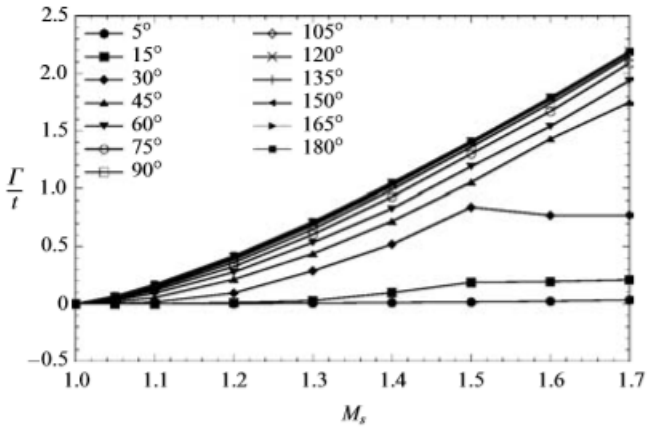


Figure 4.25: Effect of incident shocks in circulation rate (source: (Sun and Takayama, 2002))

D Finite volume discretization

One important characteristic of compressible flows is that its properties are not only transported by the flow, but also by the propagation of waves. Therefore, it is important that the selected solver takes into account this flow's characteristic. *rhoCentralFoam* does that by interpolating from neighbour cells to a face (Greenshields et al., 2009).

As this is an inviscid problem, there is only need to discretize convective, difusive and temporal terms, as the laplacian one of the viscosity is 0 (Marcantoni et al., 2012).

The convective terms to be discretized are $\nabla\cdot(V\rho)$, $\nabla\cdot(V(V\rho))$, $\nabla\cdot(V(E\rho))$ and $\nabla\cdot(Vp)$. As this is a compressible flow, the classical linear interpolation with upwinding to stailize the solution cannot be used (as in this case, the transport can occur in any direction). Therefore, letting Ψ be a generic variable in a convective term, and integrating over the control volume as shown in (Kurganov and Tadmor, 2000):

$$\int_V \nabla\cdot(V\Psi)dV = \sum_f (S_f\cdot V)\Psi_f \approx \sum_f \phi_f\Psi_f \tag{97}$$

in order to obtain Ψ_f, the flux is separated in two directions (Ψ_{f+} for the outward flux and Ψ_{f-} for the inward flux) and the general schema is used (Greenshields et al., 2009):

$$\sum_f \phi_f\Psi_f = \sum_f (\epsilon\phi_{f+}\Psi_{f+} + (1-\epsilon)\phi_{f-}\Psi_{f-} + w_f(\Psi_{f-} - \Psi_{f+})) \tag{98}$$

As it can be observed, the first two terms in equation 98 are flux expressions for both outward and inward directions. The third term is required for expresions like $\nabla\cdot(V(V\rho))$, where the convective term comes from a substantial derivative. In this

specific case, the value of = 0.5, as the flux type used is the depicted by (Kurganov and Tadmor, 2000), which is a central upwind scheme.

Finally, volumetric fluxes can be calculated as follows:

$$\Psi_{f+} = max(c_{f+}|S_f| + \phi_{f+}, c_{f-}|S_f| + \phi_{f-}, 0) \tag{99}$$

$$\Psi_{f-} = max(c_{f+}|S_f| - \phi_{f+}, c_{f-}|S_f| - \phi_{f-}, 0) \tag{100}$$

where $c_{f\pm}$ is the speed of the sound at each face. To complete the terms in 98, the diffussive volumetric flux term w_f can be computed as $\epsilon max(\Psi_{f+}, \Psi_{f-})$ for the Kurganov and Tadmor scheme.

In order to switch to higher order schemes, the interpolation procedure uses a flux limiter function called $\beta(r)$, where r is the ratio of successive gradients of the interpolated variable. For $\beta = 0$, it can be obtained upwind interpolation, and $\beta = 1$ gives linear interpolation. Another popular function is the so called VanLeer, whose function is $\beta(r) = \frac{r+|r|}{1+r}$. Their influence in the results is discused in the case section (Greenshields et al., 2009).

Using this definition, the outward flux Ψ_{f+} is evaluated as:

$$\Psi_{f+} = (1 - \beta(1 - w_f))\Psi_P + \beta(1 - w_f)\Psi_N \tag{101}$$

here, the weighting coefficient is expressed as $w_f = \frac{|S_f d_{fN}|}{|S_f d_{PN}|}$, where d_{PN} and d_{fN} connect both centroids and the face centre and the neighbour cell centre, respectively, as shown in figure 4.26

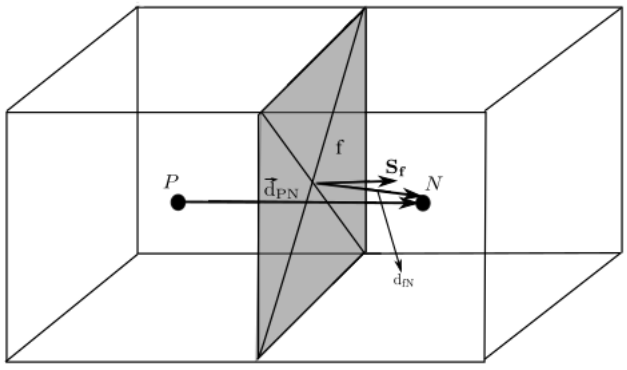


Figure 4.26: Interpolation of outward flux (source: (Greenshields et al., 2009))

Following the convection terms, the gradient terms of the governing Euler equations (in this cas ∇p) are discretised as follows:

$$\int_V \nabla p dV = \int_S dS\Psi \approx \sum_f S_f \Psi_f \tag{102}$$

For compressible flows, equation 102 is calculated as follows (the outward and inward fluxes use the interpolation mentioned before):

$$\sum_f S_f \Psi_f = \sum_f (\epsilon S_f \Psi_{f+} + (1-\epsilon) S_f \Psi_{f-}) \tag{103}$$

it can be recalled that this equation has an equivalent form as the equation 98 but without the last term, as gradient terms do not present substantial derivative dependent terms.

Finally, temporal terms are discretized using a first order explicit Euler scheme, as follows:

$$\int_V \frac{\partial(\Psi)}{\partial t} = \frac{dV(\Psi^{n+1} - \Psi^n)}{\delta t} \tag{104}$$

where δt is the time step, n is the present time step and $n+1$ is the next time step.

Following this method, two solvers are going to be explained: *rhoCentralFoam* and *sonicFoam*, which will be the one that are going to be used in this thesis

D.1 rhoCentralFoam solver

Regarding the inviscid forward step case in (Greenshields et al., 2009), the momentum and energy equations are solved using a time-splitting approach, where the Euler equations are solved explicitly.

The solution starts withe the inward and outward cell face values of ρ, T and $\hat{u}$ (where, from now on, $\hat{x} = \rho x$). This values are interpolated from cell centres's values using the selected limiter and then introduced in the convective discretization mentioned above.

With this values, the value of ρ is calculated from the continuity equation (Equation 47). After that, the value of $\hat{v}$ is calculated from the momentum Equation 105 (this value is usually corrected afterwards in a viscous apporach):

$$\frac{\delta \hat{V}}{\delta t} + \nabla \cdot (V(\hat{V})) + \nabla p = 0 \tag{105}$$

where the velocity v is updated as $v = \frac{(\hat{V})}{\rho}$.

Regarding the energy equation, the procedure is identical. An inviscid predictor $\hat{E}$ is calculated using the equation49:

$$\frac{\hat{E}}{\Delta t} + \nabla \cdot (V(\hat{E})) + \nabla \cdot (Vp) = 0 \tag{106}$$

and then, $E = \frac{\hat{E}}{\rho}$. After that, with the values of V, ρ and $\hat{E}$, the temperature is calcualted:

$$T = \frac{1}{c_v}\left(\frac{\hat{E}}{\rho} - \frac{|v^2|}{2}\right) \tag{107}$$

Equation 107 is of extreme importance because if the flux limiter function is not properly defined, in the first calculations the temperature can fall below 0, resulting in a divergence of the method. This issue has been encountered during the simulations and will be explained later.

Finally, in order to calculate all the necessary variables, the preessure p is updated using the ideal gas equation of state (Equation 50).

D.2 sonicFoam solver

It is a transient transonic/supersonic solver,suitable for compressible turbulent gases. It is based in the PISO algorithm and uses a pressure based approach, the most recommendable for subsonic flows (in this case, it is going to be used in the transonic region).

According to (Marcantoni et al., 2012), the required equation for pressure is derived from the momentum and the continuity equations. This procedure can be inconsistent because of the difficulty of discretizing all the terms in a consistent manner with the discretization of these terms in the parent equations. This can result in a velocity field which may not satisfy both momentum and continuity equation. Notwithstanding, the pressure equation can be derived from the discrete forms of the momentum and continuity equations (see the method depicted in (Issa, 1981)).

Therefore, the governing equations are expressed in finite difference form using the Euler implicit difference scheme:

$$\frac{1}{\delta t}(\rho^{n+1} - \rho^n) + \Delta_i(\rho V_i)^{n+1} = 0 \tag{108}$$

$$\frac{1}{\delta t}((\rho V_j)^{n+1} - (\rho V_j)^n) = Hu^{n+1} - \Delta_i p^{n+1} \tag{109}$$

$$\frac{1}{\delta t}((\rho E)^{n+1} - (\rho E)^n) = GE^{n+1} - \Delta_i p V_i^{n+1} \tag{110}$$

where n and $n+1$ are the current time step and the successive one and Δ_i is the spatial discretization corresponding to $\frac{\delta}{\delta x_i}$.

H defines, in this case, the finite difference of the convective fluxes of momentum, G is the finite difference of the convective fluxes of energy. These coefficients can be expressed as

$$H(V_i) = A_g V_i, g \tag{111}$$

$$G(E) = B_g E_g \tag{112}$$

where the suffix g is an identifier of each grid point an takes into account all the contribution of each surrounding node. A and B are functions of density and velocity. The coefficients shown in Equations 112 and 111 should be linearised. These coefficients are assumed constant over δt. An extra term S, regarding the diffusive effects of the equations of momentum and energy, should be added, but since this is an invsicid scheme, it will be ignored.

One important fact about this solver is that it requires the internal energy to be discretised and solved, as well
Like in any PISO scheme, the conservation of mass is accomplished by a predictor step, as follows.

D.3 A three stage scheme

D.3.1 Momentum predictor step

First of all, the equation of momentum is solved implicitly, using the pressure and density of the previous state, as stated in Equation 113 (Issa, 1981):

$$\left(\frac{1}{\delta t} - \frac{A_0}{\rho^n}\right)\rho^n V_i^* = HV_i^* - \Delta_i p^n + \frac{\rho^n V_i^n}{\delta t} \tag{113}$$

yielding the solution of V_i^*.

D.3.2 First momentum corrector step

In order to correct the preliminary value of V_i*, the momentum equation is presented in its explicit form using this former value:

$$\left(\frac{1}{\delta t} - \frac{A_0}{\rho^n}\right)\rho^* V_i^{**} = HV_i^* - \Delta_i p^* + \frac{\rho^n V_i^n}{\delta t} \tag{114}$$

If Equation 113 is substracted from Equation 114, the momentum equation can be written in its incremental form:

$$\rho^* V_i^{**} - \rho^n V_i^* = -\left(\frac{1}{\delta t} - \frac{A_0}{\rho^n}\right)^{-1} \Delta_i (p^* - p^n) \tag{115}$$

Retaking Equation 108 as:

$$\Delta_i(\rho^* V_i^{**}) = -\frac{1}{\delta t}(\rho^* - \rho^n) \tag{116}$$

and introducing this Equation 116 into Equation 115:

$$\Delta_i \left(\left(\frac{1}{\delta t} - \frac{A_0}{\rho^n}\right)^{-1} \Delta_i \right) (p^* - p^n) = \Delta_i(\rho^n V_i^*) + \frac{1}{\delta t}(\rho^* - \rho^n) \tag{117}$$

a combined equation of ρ and p. Therefore, the density variable must be substituted by the pressure. This can be done using the equation of state:

$$\rho^* = p^* \phi(p^n, T^n) \tag{118}$$

Finally, introducing Equation 118 into Equation 117 yields the final pressure-increment equation (Issa, 1981):

$$\left(\Delta_i \left(\left(\frac{1}{\delta t} - \frac{A_0}{\rho^n}\right)^{-1} \Delta_i \right) - \frac{\phi(p^n, T^n)}{\delta t} \right) (p^* - p^n) = \Delta_i(p^n u_i^*) \tag{119}$$

When the pressure is calculated, Equation 118 can be used to calculate ρ^* and Equation 115 to calculate V_i^{**}.

D.3.3 Energy predictor step

Recalling Equation 110 and writing it in implicit form

$$\left(\frac{1}{\delta t} - \frac{B_0}{\rho^*}\right) \rho^* e^* = G'(e^*) - \Delta_i(p^* V_i^{**}) + \frac{\rho^n e^n}{\delta t} \tag{120}$$

where the convective energy term G has been splitted into G' and B_0. The latter is the central element. The value of T^* can be evaluated now from e^*.

D.3.4 Second momentum corrector step

Following the energy prediction, the momentum equation is corrected again using:

$$\left(\frac{1}{\delta t} - \frac{A_0}{\rho^*}\right) \rho^{**} V_i^{***} = H V_i^{**} - \Delta_i p^{**} + \frac{\rho^n V_i^n}{\delta t} \tag{121}$$

Converting Equation 121 into its incremental form

$$\rho^{**}V_i^{***} - \rho^{*}V_i^{**} = \left(\frac{1}{\delta t} - \frac{A_0}{\rho^*}\right)^{-1}\left(H'(V_i^{**} - V_i^{*}) - \Delta_i(p^{**} - p^{*}) - A_0\left(\frac{\rho^* - \rho^n}{\rho^n}V_i^{**}\right)\right) \tag{122}$$

and introducing the updated form of the continuity equation

$$\Delta_i(\rho^{**}V_i^{***}) = -\frac{1}{\delta t}(\rho^{**} - \rho^n) \tag{123}$$

the second-corrector pressure equation is obtained:

$$\left(\Delta_i\left(\left(\frac{1}{\delta t} - \frac{A_0}{\rho^*}\right)^{-1}\Delta_i\right) - \frac{\phi(p^*, T^*)}{\delta t}\right)(p^{**} - p^*) = \Delta_i\left(\left(\frac{1}{\delta t} - \frac{A_0}{\rho^*}\right)^{-1}\left(H'(V_i^{**} - V_i^{*}) - A_0\frac{\rho^* - \rho^n}{\rho^n}V_i^{**}\right)\right) + \frac{p^*}{\delta t}(\phi(p^*, T^*) - \phi(p^n, T^n)) \tag{124}$$

In this final equation, apart of invoking the Equation 118, another modified version has been invoked:

$$\rho^{**} = p^{**}\phi(p^*, T^*) \tag{125}$$

Equation 125 is used to evaluate ρ^{**}, while Equations 124and 122 yield p^{**} and V_i^{**}, respectively. With the previously calculated T^*, a two-stage scheme will jumpt to the next time step $n+1$. In this case, however, a third stage is added for more accuracy in the results (Issa, 1981). Therefore, variables T^*, V_i^{***}, p^{**} and ρ^{**} need to be updated.

D.3.5 Energy corrector step

Using and explicit expression

$$\left(\frac{1}{\delta t} - \frac{B_0}{\rho^{**}}\right)\rho^{**}e^{**} = G'(e^*) - \Delta_i(p^{**}V_i^{***}) + \frac{\rho^n e^n}{\delta t} \tag{126}$$

and extracting from Equation 126 the Equation 127, the incremental energy equation is obtained:

$$\left(\frac{1}{\delta t} - \frac{B_0}{\rho^{**}}\right)\rho^{**}e^{**} - \left(\frac{1}{\delta t} - \frac{B_0}{\rho^{*}}\right)\rho^{*}e^{*} = -\Delta_i(V_i^{***}p^{**} - p^*V_i^{**}) \tag{127}$$

Again, $T**$ is obtained from e^{**} and V_i^{**} and the definition of a perfect gas.

D.3.6 Third momentum corrector step

Finally, and in a similar way like the first and second corrector steps, the momentum equation is written as (Issa, 1981):

$$\left(\frac{1}{\delta t}-\frac{A_0}{\rho^{**}}\right)\rho^{***}V_i^{****}=H'V_i^{**}-\Delta_i p^{***}+\frac{\rho^n V_i^n}{\delta t} \tag{128}$$

and in incremental form:

$$\rho^{***}V_i^{****}-\rho^{**}V_i^{***}=\left(\frac{1}{\delta t}-\frac{A_0}{\rho^{**}}\right)^{-1}\left(-\Delta_i(p^{***}-p^{**})-A_0\left(\frac{\rho^{**}-\rho^n}{\rho^n}V_i^{***}\right)\right) \tag{129}$$

Finally, using the continuity relation

$$\Delta_i(\rho^{***}V_i^{****})=-\frac{1}{\delta t}(\rho^{***}-\rho^n) \tag{130}$$

and the modified equation of state

$$\rho^{***}=p^{***}\phi(p^{**},T^{**}) \tag{131}$$

the pressure equation is obtained:

$$\begin{aligned}\left(\Delta_i\left(\left(\frac{1}{\delta t}-\frac{A_0}{\rho^{**}}\right)^{-1}\Delta_i\right)-\frac{\phi(p^{**},T^{**})}{\delta t}\right)(p^{***}-p^{**})=\Delta_i\Bigg(-\left(\frac{1}{\delta t}-\frac{A_0}{\rho^{**}}\right)^{-1}A_0\\ \left(\frac{\rho^{**}-\rho^n}{\rho^n}V_i^{***}\right)\Bigg)+\frac{p^{**}}{\delta t}(\phi(p^{**},T^{**})-\phi(p^{*},T^{*}))\end{aligned} \tag{132}$$

and where the values of p^{***}, V_i^{****} and ρ^{***} are obtained using their equivalent relations as stated in first and second stage and which compute the final values of the solutions of Equations 108, 109 and 110 (Issa, 1981).

E Snatch force and snatch velocity calculation

The procedure that is going to be explained follows the development shown in the report of (Solt and Ria, 1961).

One of the most important stages in a parachutes deployment is the line stretch phase. Due to the liberation of the canopy, a differential deceleration rate is created between the canopy and the suspended load. The force that equals these velocities is the snatch force, and the resultant velocity is known as snatch velocity.

In order to calculate both the snatch force and velocity, (Solt and Ria, 1961) propposes to solve both the energy and velocity equations during the parachute's deployment.

E.1 Energy equation

It is possible to calculate the snatch force from the velocity equation. during the parachute's inflation it gains a relative velocity with respect the main body. The maximum of this velocity is achieved when the suspension-lines are fully elongated at L_{sn}, without strecthing. Finally, when the lines get stretched at L_{Max}, the canopy its accelerated to the velocity of the primary body, thus reducing the relative velocity to 0 (see Figure 4.27).

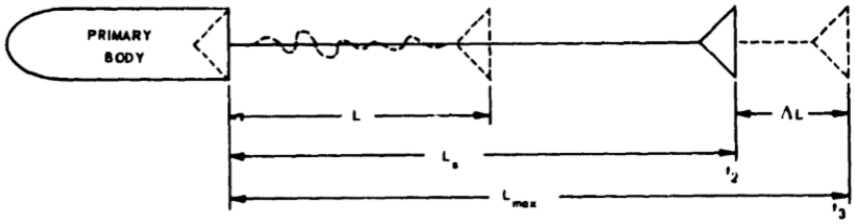


Figure 4.27: Deployment sequence (source: (Solt and Ria, 1961))

Therefore, this strain energy can be computed as:

$$\Delta E = \frac{1}{2} m_c (V_{rel,L_{sn}}^2 - V_{rel,L_{Max}}^2) \tag{133}$$

This increment of energy is transmitted through the suspension lines. As the $V_{rel,L_{Max}} = 0$ because is the velocity at maximum elongation, the force P transmitted through the elongation lines can be written as

$$\Delta E = \int_{L_{sn}}^{L} {}_{Max} P dL \longrightarrow \frac{1}{2} m_c (V_{rel,L_s}^2 - V_{rel,L_{Max}}^2) = \int_{L_s}^{L} {}_{Max} P dL \tag{134}$$

where the force P can be expressed as

$$P = \frac{N \sigma_{maxlines} \xi}{\xi'} \tag{135}$$

In order to properly integrate Equation 134, dL must be expressed with force P parameters. Therefore, regarding Figure 4.27

$$L_{Max} = L_{sn} + \Delta L = L_{sn} + \xi L_{sn} \tag{136}$$

dL becomes

$$dL = L_{sn} d\xi \tag{137}$$

introducing Equations 135 and 137 into 134, the result of the integral can be expressed as

$$\Delta E = \frac{1}{2} L_{sn} \frac{N \sigma_{maxlines} \xi^2}{\xi'} + K \tag{138}$$

where K is the integration constant which can be calculated from initial values (at $t = 0$, $E = \Delta E = 0$. Its value is 0. It is possible to calculate the percent of elongation ξ from Equation 143 as

$$\xi = \sqrt{\frac{2 \Delta E \xi'}{N L_{sn} P'}} \tag{139}$$

And finally, substituing ΔE in Equation 139 and introducing the result into 135, the snatch force can be expressed as a fraction of the $V_{rel,L_{sn}}$ (or snatch velocity v_{sn}) as

$$P = \sqrt{\frac{m_c V_{rel,L_{sn}}^2 N P'}{L_{sn} \xi'}} \longrightarrow F_s = \sqrt{\frac{m_c v_{sn}^2 N P'}{L_{sn} \xi'}} \tag{140}$$

Therefore, the only parameter that needs to be calculated is v_{sn}.

E.2 Velocity equation

As mentioned before, the snatch velocity can be calculated from the velocity equation. This velocity is the one that the system has whene the canopy starts inflating and the snatch force has just occurred (Solt and Ria, 1961).

In order to calculate this velocity, it is supposed that the time for the lines to stretch is negligible and that the snatch velocity will be the one at the beggining of the inflation. All the procedure supposes that the deployment occurs in an horizontal plane.

Therfore, using Newton's Second Law

$$m \frac{dv}{dt} = -\frac{1}{2} \rho C_D S v^2 \tag{141}$$

where, if $J = \frac{\rho C_D S}{2m}$ is defined, yields

$$\frac{dv}{v^2} = -J dt \tag{142}$$

integrating

$$\frac{1}{v} = Jt + K \tag{143}$$

where, supposing that at $t = 0$, $v = v_0$:

$$K = \frac{1}{v_0} \tag{144}$$

which yields the following expression for the suspension lines velocity

$$v = \frac{v_0}{Jv_O t + 1} \tag{145}$$

This velocity can also be expressed in function of the suspension lines length

$$v = \frac{dl}{dt} \longrightarrow \frac{dl}{dt} = \frac{v_0}{Jv_O t + 1} \tag{146}$$

Integration of Equation 146 results

$$l = \frac{1}{J} \ln 1 + Jv_0 t \tag{147}$$

where the constant of integration is 0 as at the beginning of the inflation, $l = 0$. Assuming that, in order to extend the suspension lines the distance L_1 it takes t_1 seconds:

$$l_1 = \frac{1}{J_1} \ln 1 + J_1 v_0 t_1 \tag{148}$$

In Equation 148, $J_1 = \frac{\rho(C_D S)_b ody}{2(m_b+m_c)}$ as the only component of the system that produces drag is the body (the canopy is supposed to be uninflated). Regarding the launching vehicle, during t_1 it travels the distance $l_2 = v_0 t_1$. Therefore, the distance between the launching vehicle and the decelerating system can be written as

$$l_2 - l_1 = v_0 t_1 - \frac{1}{J_1} \ln 1 + J_1 v_0 t_1 \tag{149}$$

Equation 149 accounts only for the horitzontal traveled distance. Assuming that in the vertical axis the traveled distance equals to $\frac{1}{2}gt^2$, the total distance is the contribution of each factor as

$$L_1 = \sqrt{\frac{g^2 t_1^4}{4} + \left(v_0 t_1 - \frac{1}{J_1} \ln 1 + J_1 v_0 t_1 \right)^2} \tag{150}$$

Once Equation 150 is solved, the velocity at the extension of suspension lines can be determined using Equation 145:

$$v_d = \frac{v_0}{J_1 v_0 t_1 + 1} \tag{151}$$

With all this data, the only parameter that is needed is the time t_2 to extend the suspension lines from its full extension to the moment before they strech (at t_2, the velocity of the system is the snatch velocity).

In a similar way as in Equation 148, the horisontal distance traveled by the primary body can be written as

$$l_b = \frac{1}{J_b} \ln 1 + J_b v_d t_2 \tag{152}$$

where $J_b = \frac{\rho (C_D S)_b}{2m_b}$. For the secondary body

$$l_c = \frac{1}{J_c} \ln 1 + J_c v_d t_2 \tag{153}$$

where $J_c = \frac{\rho (C_D S)_c}{2m_c}$. The separation between the two bodies can be accounted as

$$L_{sn} = l_b - l_c = \frac{1}{J_b} \ln 1 + J_b v_d t_2 - \frac{1}{J_c} \ln 1 + J_c v_d t_2 \tag{154}$$

where the time t_2 can be determined. With this time the velocities of both the primary and secondary bodies can be calculated. As it is assumed that the time between stretch and snatch is small, the deceleration of the primary body can be neglected, which yields a snatch velocity of

$$v_{sn} = \frac{v_d}{J_c v_d t_2 + 1} \tag{155}$$

F Aerodynamic coefficients

In this section all the analytical functions used in the flight simulator are presented.

F.1 Transonic, supersonic and hypersonic phase

Two different models, one for each geometry, are presented. The AoA is taken in [°]. The curves are fitted using the method of the least squares

F.1.1 70°aeroshell

The first model corresponds to lift-drag 2D model:
0.75 < M < 2:

$$\begin{aligned} C_D = & C_{D0}(12.0418 - 15.0510M + 7.4548M^3 - 2.511M^4) + \\ & C_{D\alpha}(1.57 \cdot 10^{-2} - 8.7 \cdot 10^{-3}M)\alpha + \\ & C_{D\alpha^2}(-2.57 \cdot 10^{-2} + 5.86 \cdot 10^{-2}M - 4.38 \cdot 10^{-2}M^2 + 1.06 \cdot 10^{-2}M^3)\alpha^2 \end{aligned} \tag{156}$$

$$C_L = C_{L\alpha}(8.19 \cdot 10^{-2} - 6.79 \cdot 10^{-2}M + 2.1 \cdot 10^{-2}M^2)\alpha \tag{157}$$

$$C_M = C_{M\alpha}(2.08 \cdot 10^{-2} - 5.33 \cdot 10^{-2}M + 4.13 \cdot 10^{-2}M^2 - 1.02 \cdot 10^{-2}M^3)\alpha \tag{158}$$

M > 2:

$$\begin{aligned} C_D =& C_{D0}(1.7893 - 9.3 \cdot 10^{-3}M + 5.605 \cdot 10^{-4}M^2 - 2.8403 \cdot 10^{-5}M^3 + \\ & 6.7132 \cdot 10^{-7}M^4) + C_{D\alpha}(-1.7291 \cdot 10^{-4} + 6.1487 \cdot 10^{-5}M - \\ & 1.381 \cdot 10^{-5}M^2 + 2.8697 \cdot 10^{-6}M^3 - 2.4111 \cdot 10^{-7}M^4 + 8.5715 \cdot 10^{-9}M^5 \\ & - 1.0911 \cdot 10^{-10}M^6)\alpha + C_{D\alpha^2}(-2.1054 \cdot 10^{-4} - 1.008 \cdot 10^{-4}M + \\ & 5.6816 \cdot 10^{-6}M^2 - 1.051 \cdot 10^{-7})\alpha^2 \end{aligned} \tag{159}$$

$$\begin{aligned} C_L =& C_{L\alpha}(2.85 \cdot 10^{-2} - 1.3 \cdot 10^{-3}M + 1.1937 \cdot 10^{-4}M^2 - 5.0317 \cdot 10^{-6}M^3 + \\ & 7.9215 \cdot 10^{-8}M^4)\alpha \end{aligned} \tag{160}$$

$$\begin{aligned} C_M =& C_{M\alpha}(9.0693 \cdot 10^{-4} - 1.9 \cdot 10^{-3}M + 3.8195 \cdot 10^{-4}M^2 - 3.9286 \cdot 10^{-5}M^3 + \\ & 2.1816 \cdot 10^{-6}M^4 - 6.1676 \cdot 10^{-8}M^5 + 6.9441 \cdot 10^{-10}M^6)\alpha \end{aligned} \tag{161}$$

The second model corresponds to the ballistic descent:

0.75 < M < 1.5:

$$C_D = C_{D0}(1.6095 + 5.03 \cdot 10^{-2}M) \tag{162}$$

1.5 < M < 5:

$$C_D = C_{D0}(1.8236 - 6.23 \cdot 10^{-2}M) \tag{163}$$

M > 5:

$$C_D = C_{D0}(1.7436 - 1.28 \cdot 10^{-2}M + 3.6975 \cdot 10^{-4}M^2) \tag{164}$$

F.1.2 45°aeroshell

The first model corresponds to lift-drag 2D model:

0.75 < M < 2:

$$\begin{aligned} C_D =& C_{D0}(11.5051 - 14.4039M + 7.0865M^3 - 2.3811M^4) + \\ & C_{D\alpha}(3.3 \cdot 10^{-3} - 5.9 \cdot 10^{-3}M)\alpha + \\ & C_{D\alpha^2}(-7.5 \cdot 10^{-3} + 1.44 \cdot 10^{-2}M - 9.4 \cdot 10^{-3}M^2 + 2 \cdot 10^{-3}M^3)\alpha^2 \end{aligned} \tag{165}$$

$$C_L = C_{L\alpha}(5.11 \cdot 10^{-2} - 4.13 \cdot 10^{-2}M + 1.18 \cdot 10^{-2}M^2)\alpha \tag{166}$$

$$C_M = C_{M\alpha}(1.83 \cdot 10^{-2} - 4.81 \cdot 10^{-2}M + 3.75 \cdot 10^{-2}M^2 - 9.3 \cdot 10^{-3}M^3)\alpha \tag{167}$$

M > 2:

$$\begin{aligned} C_D =& C_{D0}(1.8362 - 1.338 \cdot 10^{-1}M + 1.33 \cdot 10^{-2}M^2 - 5.812 \cdot 10^{-4}M^3 + \\ & 9.1857 \cdot 10^{-6}M^4) + C_{D\alpha}(-1.1 \cdot 10^{-3} + 5.6446 \cdot 10^{-4}M - \\ & 6.7057 \cdot 10^{-5}M^2 - 6.5375 \cdot 10^{-7}M^3 + 5.4965 \cdot 10^{-7}M^4 - 3.1346 \cdot 10^{-8}M^5 \\ & + 5.3406 \cdot 10^{-10}M^6)\alpha + C_{D\alpha^2}(-1.2 \cdot 10^{-3} + 3.7033 \cdot 10^{-4}M - \\ & 2.4041 \cdot 10^{-5}M^2 + 4.8805 \cdot 10^{-7})\alpha^2 \end{aligned} \tag{168}$$

$$C_L = C_{L\alpha}(2.55 \cdot 10^{-2} - 9.2 \cdot 10^{-3}M + 1.1 \cdot 10^{-2}M^2 - 5.3698 \cdot 10^{-5}M^3 + 8.8645 \cdot 10^{-7}M^4)\alpha \tag{169}$$

$$C_M = C_{M\alpha}(-1.7901 \cdot 10^{-4} - 10^{-3}M + 1.15 \cdot 10^{-4}M^2 - 6.6574 \cdot 10^{-7}M^3 - 5.9762 \cdot 10^{-7}M^4 + 3.4115 \cdot 10^{-8}M^5 - 5.6548 \cdot 10^{-10}M^6)\alpha \tag{170}$$

The second model corresponds to the ballistic descent:
0.75 < M < 1.5:

$$C_D = C_{D0}(0.0601 + 9.506 \cdot 10^{-1}M) \tag{171}$$

1.5 < M < 5:

$$C_D = C_{D0}(1.7524 - 1.1808 \cdot 10^{-1}M) \tag{172}$$

M > 5:

$$C_D = C_{D0}(1.1931 - 1.86 \cdot 10^{-2}M + 5.0868 \cdot 10^{-4}M^2) \tag{173}$$

F.1.3 Parachute

The equation of the parachute corresponds to a ballistic descent.
M < 0.75:

$$C_D = C_{D0}(0.6299 - 5.688M + 5.552M^2 - 24.459M^3 + 52.403M^4 - 52.555M^5 + 19.384M^6) \tag{174}$$

G OpenFOAM case files

In order to make feasible the development of the explained cases, in this last appendix the reader can find attached the files of each case. As there are a lot of cases, only will be attached the graded mesh case for the validation, and one axisymmetric case at $M = 5$ and one transonic case at $M = 0.8$ for the aeroshell case, as all the necessary files for the other cases are contained in these formers.

G.1 Validation case files

p

```
    /*---------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  dev
|
```

```
|    \\  /    A nd           | Web:      www.OpenFOAM.org
|
|     \\/     M anipulation  |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [1 -1 -2 0 0 0 0];

internalField   uniform 180.0431;

boundaryField
{
    wedge0
    {
        type            wedge;
    }
    ghv
    {
        type            zeroGradient;
    }
    outlet
    {
        type            waveTransmissive;
        value           uniform 180.0431;
        field           p;
        gamma           1.4;
        lInf            1.9;
        fieldInf        180.0431;
    }
    inlet
    {
        type            fixedValue;
        value           uniform 180.0431;
    }
```

```
    wedge1
    {
        type            wedge;
    }
    top
    {
        type            zeroGradient;
    }
}


// ************************************************************************* //
```

T

```
    /*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  dev
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      T;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 1 0 0 0];

internalField   uniform 47.7124;

boundaryField
{
    wedge0
```

```
    {
        type            wedge;
    }
    ghv
    {
        type            zeroGradient;
    }
    outlet
    {
        type            zeroGradient;
    }
    inlet
    {
        type            fixedValue;
        value           uniform 47.7124;
    }
    wedge1
    {
        type            wedge;
    }
    top
    {
        type            zeroGradient;
    }
}


// ************************************************************************* //
```

U

```
    /*---------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  dev
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------------*/
FoamFile
```

```
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    location    "0";
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (-1401.203 0 0);

boundaryField
{
    wedge0
    {
        type            wedge;
    }
    ghv
    {
        type            slip;
    }
    outlet
    {
        type            advective;
        phi             phi;
        lInf            1.9;
        fieldInf        (-1401.203 0 0);
    }
    inlet
    {
        type            fixedValue;
        value           uniform (-1401.203 0 0);
    }
    wedge1
    {
        type            wedge;
    }
    top
    {
        type            zeroGradient;
    }
```

```
}


// ************************************************************************* //
```

thermophysicalProperties

```
    /*---------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  dev
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      thermophysicalProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

thermoType
{
    type            hePsiThermo;
    mixture         pureMixture;
    transport       const;
    thermo          hConst;
    equationOfState perfectGas;
    specie          specie;
    energy          sensibleEnthalpy;
}

mixture
{
    // air
    specie
```

```
    {
        nMoles          1;
        molWeight       28.96;
    }
    thermodynamics
    {
        Cp              1004;
        Hf              2.54e6;
    }
    transport
    {
        mu              0;
        Pr              0.7;
    }
}


// ************************************************************************* //
```

turbulenceProperties

```
    /*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  dev
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      turbulenceProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

simulationType  laminar;
```

```
// ************************************************************************* //
```

blockMeshDict

```
    /*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  dev
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

convertToMeters 0.001;

// Geometrical parameters
halfAngle #calc "degToRad(2.5)";
c350 #calc "cos($halfAngle)*350";
s350 #calc "sin($halfAngle)*350";
c73 #calc "cos($halfAngle)*73.66";
s73 #calc "sin($halfAngle)*73.66";
c31 #calc "cos($halfAngle)*31.75";
s31 #calc "sin($halfAngle)*31.75";

ns350 #calc "sin(-$halfAngle)*350";
ns73 #calc "sin(-$halfAngle)*73.66";
ns31 #calc "sin(-$halfAngle)*31.75";

// Element grading parameters
RefPara 1.5;
```

```
ExtraEl 1;

X1 #calc "$RefPara*12";
Z1 #calc "$RefPara*74";

X2 #calc "$RefPara*14";
Z2 #calc "$RefPara*74";

X3 #calc "$ExtraEl*$RefPara*92";
Z3 #calc "$RefPara*74";

X4 #calc "$RefPara*14";
Z4 #calc "$RefPara*24";

X5 #calc "$ExtraEl*$RefPara*92";
Z5 #calc "$RefPara*24";

X6 #calc "$ExtraEl*$RefPara*92";
Z6 #calc "3.5*$RefPara*24";

X7 #calc "4*$ExtraEl*$RefPara*24";
Z7 #calc "$RefPara*74";

X8 #calc "$ExtraEl*$RefPara*92";
Z8 #calc "$RefPara*74";

vertices
(
        // Main control volume
        (800 0 0)
        (0 $s350 $c350)
        (0 $ns350 $c350)
        (800 $ns350 $c350)
        (800 $s350 $c350)

        (0 0 0)
        // GHV
        (501.6 0 0)
        (501.6 $s73 $c73)
        (501.6 $ns73 $c73)
        (400 $s73 $c73)

        (400 $ns73 $c73)
```

```
        (400 $s31 $c31)
        (400 $ns31 $c31)
        (300 $s31 $c31)
        (300 $ns31 $c31)

        (300 0 0)
        // Auxiliary points: refining surfaces
        (501.6 $s350 $c350)
        (501.6 $ns350 $c350)
        (400 $s350 $c350)
        (400 $ns350 $c350)

        (300 $s350 $c350)
        (300 $ns350 $c350)

        (0 $s73  $c73)
        (0 $ns73  $c73)
        (800 $s73 $c73)

        (800 $ns73 $c73)
        (300 $s73 $c73)
        (300 $ns73 $c73)
        (0 $s31 $c31)
        (0 $ns31 $c31)
);

blocks
( // Ordered from lower Z to higher Z

        // Wake
        hex (5 28 29 5 15 13 14 15) ($X1 1 $Z1) simpleGrading (1 1 1)
        hex (28 22 23 29 13 26 27 14) ($X2 1 $Z2) simpleGrading (1 1 1)
        hex (22 1 2 23 26 20 21 27) ($X3 1 $Z3) simpleGrading (10 1 1)

        // Rear-Body
        hex (13 26 27 14 11 9 10 12) ($X4 1 $Z4) simpleGrading (1 1 1)
        hex (26 20 21 27 9 18 19 10) ($X5 1 $Z5) simpleGrading (10 1 1)

        // Upper-body
        hex (9 18 19 10 7 16 17 8) ($X6 1 $Z6) simpleGrading (10 1 1)

        // Freestream
        hex (6 7 8 6 0 24 25 0) ($X7 1 $Z7) simpleGrading (1 1 10)
```

```
        hex (7 16 17 8 24 4 3 25) ($X8 1 $Z8) simpleGrading (10 1 10)
);

edges
(
        arc 3 4 (800 0 350)
        arc 17 16 (501.6 0 350)
        arc 19 18 (400 0 350)
        arc 21 20 (300 0 350)
        arc 8 7 (501.6 0 73.66)
        arc 10 9 (400 0 73.66)
        arc 12 11 (400 0 31.75)
        arc 14 13 (300 0 31.75)
        arc 2 1 (0 0 350)
);

boundary
(
    axis
    {
        type empty;
        faces
        (
                        (5 15 15 5)
                        (6 0 0 6)
        );
    }


    inlet
    {
        type patch;
        faces
        (
            (0 0 25 24)
                        (24 25 3 4)
        );
    }

    outlet
    {
        type patch;
        faces
```

```
    (
        (5 5 29 28)
                    (28 29 23 22)
                    (22 23 2 1)
    );
}

top
{
    type patch;
    faces
    (
                    (1 20 21 2)
                    (20 18 19 21)
                    (18 16 17 19)
                    (16 4 3 17)
    );
}

wedge0
{
    type wedge;
    faces
    (
        (0 25 8 6)
                    (25 3 17 8)

                    (8 17 19 10)

                    (12 10 27 14)
                    (10 19 21 27)

                    (15 14 29 5)
                    (14 27 23 29)
                    (27 21 2 23)
    );
}

wedge1
{
    type wedge;
    faces
    (
```

```
            (0 24 7 6)
                        (24 4 16 7)

                        (7 16 18 9)

                        (11 9 26 13)
                        (9 18 20 26)

                        (15 13 28 5)
                        (13 26 22 28)
                        (26 20 1 22)
        );
    }


    ghv
    {
        type patch;
        faces
        (
                        (15 15 14 13)
                        (11 12 14 13)
                        (9 11 12 10)
                        (7 8 10 9)
            (6 6 8 7)
        );
    }
);

mergePatchPairs
(
);

// ************************************************************************* //
```

controlDict

```
    /*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  dev
|
```

```
|    \\  /    A nd           | Web:      www.OpenFOAM.org
|
|     \\/     M anipulation  |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     rhoCentralFoam;

startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         0.0015;

deltaT          1e-8;

writeControl    runTime;

writeInterval   0.00001;

purgeWrite      0;

writeFormat     ascii;

writePrecision  7;

writeCompression off;

timeFormat      general;

timePrecision   6;
```

```
runTimeModifiable true;

adjustTimeStep   yes;

maxCo            0.5;

maxAlphaCo       0.5;

maxDeltaT        5e-7;


// ************************************************************************* //
```

fvSchemes

```
    /*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  dev
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

fluxScheme      Kurganov;

ddtSchemes
{
    default         Euler;
}
```

```
gradSchemes
{
    default          Gauss linear;
        grad(p)          linearUpwind phi;
}

divSchemes
{
    default          none;
        div(phi,U)              Gauss vanLeerV grad(U);
        div(phi,h)              Gauss vanLeerV phi;
        div(u,rho)              Gauss vanLeerV phi;
        div(u,p)                Gauss vanLeerV phi;
}

laplacianSchemes
{
    default          none;
}

interpolationSchemes
{
    default          linear;
    reconstruct(rho) upwind phi;
    reconstruct(U)   upwind phi;
    reconstruct(T)   upwind phi;
}

snGradSchemes
{
    default          corrected;
}

// ************************************************************************* //
```

fvSolution

```
    /*---------------------------*- C++ -*----------------------------------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  dev
|
```

```
|    \\  /    A nd           | Web:      www.OpenFOAM.org
|
|     \\/     M anipulation  |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    "(rho|rhoU|rhoE)"
    {
        solver          diagonal;
    }

    U
    {
        solver          smoothSolver;
        smoother        GaussSeidel;
        nSweeps         2;
        tolerance       1e-09;
        relTol          0;
    }

    h
    {
        $U;
        tolerance       1e-10;
        relTol          0;
    }
}


// ************************************************************************* //
```

G.2 Aeroshell case files - Transonic case

p

```
    /*---------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  dev
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [1 -1 -2 0 0 0 0];

internalField   uniform 205.51;

boundaryField
{
    back
    {
        type            empty;
    }
    ghv
    {
        type            zeroGradient;
    }
    outlet
    {
        type            waveTransmissive;
        value           uniform 205.51;
```

```
        field           p;
        gamma           1.31;
        psi             thermo:psi;
        lInf            170;
        fieldInf        205.51;
    }
    inlet
    {
        type            waveTransmissive;
        value           uniform 205.51;
        field           p;
        gamma           1.31;
        psi             thermo:psi;
        lInf            40;
        fieldInf        205.51;
    }
    front
    {
        type            empty;
    }
    top
    {
        type            zeroGradient;
    }
}


// ************************************************************************* //
```

T

```
    /*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  dev
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------------*/
FoamFile
```

```
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      T;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 1 0 0 0];

internalField   uniform 184.59;

boundaryField
{
    back
    {
        type            empty;
    }
    ghv
    {
        type            zeroGradient;
    }
    outlet
    {
        type            waveTransmissive;
        value           uniform 184.59;
        field           T;
        gamma           1.31;
        psi             thermo:psi;
        lInf            170;
        fieldInf        184.59;
    }
    inlet
    {
        type            fixedValue;
        value           uniform 184.59;
    }
    front
    {
        type            empty;
    }
    top
```

```
    {
        type            zeroGradient;
    }
}


// ************************************************************************* //
```

U

```
    /*---------------------*- C++ -*----------------------------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  dev
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    location    "0";
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (-172.29 0 0);

boundaryField
{
    back
    {
        type            empty;
    }
    ghv
    {
```

```
        type            slip;
    }
    outlet
    {
        type            waveTransmissive;
        value           uniform (-172.29 0 0);
        field           U;
        gamma           1.31;
        psi             thermo:psi;
        lInf            170;
        fieldInf        (-172.29 0 0);;
    }
    inlet
    {
        type            waveTransmissive;
        value           uniform (-172.29 0 0);
        field           U;
        gamma           1.31;
        psi             thermo:psi;
        lInf            40;
        fieldInf        (-172.29 0 0);
    }
    front
    {
        type            empty;
    }
    top
    {
        type            zeroGradient;
    }
}


// ************************************************************************* //
```

thermophysicalProperties

```
    /*---------------------------*- C++ -*----------------------------------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  dev
|
```

```
|    \\  /    A nd           | Web:      www.OpenFOAM.org
|
|     \\/     M anipulation  |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      thermophysicalProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

thermoType
{
    type            hePsiThermo;
    mixture         pureMixture;
    transport       const;
    thermo          hConst;
    equationOfState perfectGas;
    specie          specie;
    energy          sensibleInternalEnergy;
}

mixture
{
    // air
    specie
    {
        nMoles          1;
        molWeight       44.01;
    }
    thermodynamics
    {
        Cp              849;
        Hf              2.05e5;
    }
    transport
    {
        mu              0;
        Pr              0.7;
```

```
    }
}


// ************************************************************************* //
```

turbulenceProperties

```
    /*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  dev
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      turbulenceProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

simulationType  laminar;


// ************************************************************************* //
```

blockMeshDict

```
    /*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  dev
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
```

```
|     \\/     M anipulation  |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

convertToMeters 1;

// Element grading parameters
RefPara 0.9;

X1 #calc "$RefPara*40";
Z1 #calc "$RefPara*150";

X2 #calc "$RefPara*260";
Z2 #calc "$RefPara*150";

X3 #calc "$RefPara*260";
Z3 #calc "$RefPara*50";

X4 #calc "12";
Z4 #calc "$RefPara*140";

X5 #calc "$RefPara*260";
Z5 #calc "$RefPara*140";


vertices
(
        // Main control volume
        (120 -0.1 0)
        (120 0.1 80)
        (120 -0.1 80)

        (0 -0.1 0)
        (0 0.1 80)
        (0 -0.1 80)
```

```
        // GHV
        (11 -0.1 0)
        (11 0.1 0.75)
        (11 -0.1 0.75)

        (10 -0.1 0)
        (10 0.1 3)
        (10 -0.1 3)

        // Auxiliary points: refining surfaces
        (120 0.1 0.75)
        (120 -0.1 0.75)

        (11 0.1 80)
        (11 -0.1 80)

        (10 0.1 80)
        (10 -0.1 80)

        (0 0.1 3)
        (0 -0.1 3)

        (120 0.1 0)
        (0 0.1 0)
        (11 0.1 0)
        (10 0.1 0)
);

blocks
( // Ordered from lower Z to higher Z
        // Wake
        hex (21 18 19 3 23 10 11 9) ($X1 1 $Z1) simpleGrading (1 1 0.15)
        hex (18 4 5 19 10 16 17 11) ($X2 1 $Z2) simpleGrading (15 1 0.15)

        // Upper-body
        hex (10 16 17 11 7 14 15 8) ($X3 1 $Z3) simpleGrading (15 1 1)

        // Freestream
        hex (22 7 8 6 20 12 13 0) ($X4 1 $Z4) simpleGrading (1 1 150)
        hex (7 14 15 8 12 1 2 13) ($X5 1 $Z5) simpleGrading (15 1 150)
);
```

```
boundary
(
    inlet
    {
        type patch;
        faces
        (
                        (1 14 15 2)
                        (20 12 13 0)
                        (12 1 2 13)
        );
    }

    outlet
    {
        type patch;
        faces
        (
                        (3 19 18 21)
                        (19 5 4 18)
        );
    }

    top
    {
        type patch;
        faces
        (
                        (14 16 17 15)
                        (16 4 5 17)
        );
    }

    back
    {
        type empty;
        faces
        (
            (0 13 8 6)
                        (13 2 15 8)

                        (8 15 17 11)
```

```
                        (3 9 11 19)
                        (11 17 5 19)
        );
    }

    front
    {
        type empty;
        faces
        (
            (20 22 7 12)
                        (12 7 14 1)

                        (7 10 16 14)

                        (23 21 18 10)
                        (10 18 4 16)
        );
    }


    ghv
    {
        type patch;
        faces
        (
                        (6 8 7 22)
                        (7 8 11 10)
                        (23 10 11 9)
        );
    }
);

mergePatchPairs
(
);

// ************************************************************************* //
```

controlDict

```
    /*--------------------------------*- C++ -*----------------------------------*\
| =========                 |
|
```

```
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  dev
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     sonicFoam;

startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         2.5;

deltaT          7.5e-5;

writeControl    runTime;

writeInterval   0.04;

purgeWrite      0;

writeFormat     ascii;

writePrecision  7;

writeCompression off;
```

```
timeFormat      general;

timePrecision   6;

runTimeModifiable true;

adjustTimeStep  no;

maxCo           0.5;

maxAlphaCo      0.5;

maxDeltaT       1e-4;

functions
{
        #include "plotLiftDragMoment"
}


// ************************************************************************* //
```

fvSchemes

```
    /*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  dev
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
```

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default          Euler;
}

gradSchemes
{
    default          Gauss linear;
        grad(p)          Gauss linear;
}

divSchemes
{
    default          Gauss upwind;
        //div(phi,U)              Gauss vanLeerV 1;
        //div(phi,e)              Gauss vanLeer 1;
        //div(u,rho)              Gauss vanLeer 1;
        //div(u,p)                Gauss vanLeer 1;

        div(((rho*nuEff)*dev2(T(grad(U))))) Gauss linear;
        //div(phiv,p)             Gauss limitedLinear 1;
        //div(phid,p)             Gauss limitedLinear 1;
        //div(phi,K)              Gauss upwind;
}

laplacianSchemes
{
    default          none;
    laplacian((rho*nuEff),U)    Gauss linear limited corrected 0.5;
    laplacian(thermo:alpha,e)   Gauss linear limited corrected 0.5;
    laplacian(rhorAUf,p)        Gauss linear limited corrected 0.5;
}

interpolationSchemes
{
    default          linear;
    reconstruct(rho) upwind phi;
    reconstruct(U)   upwind phi;
    reconstruct(T)   upwind phi;
}
```

```
snGradSchemes
{
    default         corrected;
}

// ************************************************************************* //
```

fvSolution

```
    /*---------------------------*- C++ -*----------------------------------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  dev
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    "(p|rho)"
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-6;
        relTol          0;
    }

    "(p|rho)Final"
    {
        $p;
```

```
        relTol          0;
    }

    "(U|e)"
    {
        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-6;
        relTol          0;
    }

    "(U|e)Final"
    {
        $U;
        relTol          0;
    }
}

PIMPLE
{
    nCorrectors       3;
    nNonOrthogonalCorrectors 3;
}


// ************************************************************************* //
```

mirrorMeshDict

```
    /*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  dev
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
```

```
    format      ascii;
    class       dictionary;
    object      mirrorMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

planeType           pointAndNormal;

pointAndNormalDict
{
    basePoint       (0 0 0);
    normalVector    (0 0 1);
}

planeTolerance      1e-6;

// ************************************************************************* //
```

plotLiftDragMoment

```
    /*--------------------------------------------------------------------
/*
/* This fucntion calculates the aerodynamic forces and moments of any case.
/*
/* Source: https://www.hpc.ntnu.no/display/hpc/OpenFOAM+-+Airfoil+Calculati
/* Just by specifing the parameters and the directions, OpenFOAM has a libra
/*
/*--------------------------------------------------------------------

forces
{
    type                forces;
    functionObjectLibs  ("libforces.so");

    patches             (ghv);
    pName               p;
    UName               U;
    log                 true;

    CofR                (10.75 0 0);

    outputControl       timeStep;
    outputInterval      100;
}
```

```
forceCoeffs
{
    type                 forceCoeffs;
    functionObjectLibs   ("libforces.so");

    patches              (ghv);
    p                    p;
    U                    U;
    rho                  rhoInf;
    log                  true;

    CofR                 (10.75 0 0);
    liftDir              (0 0 -1);
    dragDir              (-1 0 0);
    pitchAxis            (0 -1 0);

    rhoInf               0.0058;
    magUInf              172.29;
    lRef                 6;
    Aref                 1.2;

    outputControl        timeStep;
    outputInterval       100;
}
/*************************************************************************
```

G.3 Aeroshell case files - Axisymmetric case

p

```
    /*--------------------*- C++ -*----------------------------*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  dev
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
```

```
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [1 -1 -2 0 0 0 0];

internalField   uniform 35.85;

boundaryField
{
    back
    {
        type            wedge;
    }
    ghv
    {
        type            zeroGradient;
    }
    outlet
    {
        type            waveTransmissive;
        value           uniform 35.85;
        field           p;
        gamma           1.31;
        psi             thermo:psi;
        lInf            170;
        fieldInf        35.85;
    }
    inlet
    {
        type            fixedValue;
        value           uniform 35.85;
    }
    front
    {
        type            wedge;
```

```
    }
    top
    {
        type            zeroGradient;
    }
}


// ************************************************************************* //
```

T

```
    /*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  dev
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      T;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 1 0 0 0];

internalField   uniform 181.9;

boundaryField
{
    back
    {
        type            wedge;
    }
```

```
    ghv
    {
        type            zeroGradient;
    }
    outlet
    {
        type            zeroGradient;
    }
    inlet
    {
        type            fixedValue;
        value           uniform 181.9;
    }
    front
    {
        type            wedge;
    }
    top
    {
        type            zeroGradient;
    }
}


// ************************************************************************* //
```

U

```
    /*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  dev
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
```

```
    class       volVectorField;
    location    "0";
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (-1068.92 0 0);

boundaryField
{
    back
    {
        type            wedge;
    }
    ghv
    {
        type            slip;
    }
    outlet
    {
        type            advective;
        phi             phi;
        lInf            170;
        fieldInf        (-1068.92 0 0);
    }
    inlet
    {
        type            fixedValue;
        value           uniform (-1068.92 0 0);
    }
    front
    {
        type            wedge;
    }
    top
    {
        type            zeroGradient;
    }
}
```

```
// ************************************************************************* //
```

thermophysicalProperties

```
    /*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  dev
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      thermophysicalProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

thermoType
{
    type            hePsiThermo;
    mixture         pureMixture;
    transport       const;
    thermo          hConst;
    equationOfState perfectGas;
    specie          specie;
    energy          sensibleInternalEnergy;
}

mixture
{
    // air
    specie
    {
        nMoles          1;
        molWeight       44.01;
```

```
    }
    thermodynamics
    {
        Cp              849;
        Hf              2.05e5;
    }
    transport
    {
        mu              0;
        Pr              0.7;
    }
}


// ************************************************************************* //
```

turbulenceProperties

```
    /*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  dev
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      turbulenceProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

simulationType  laminar;


// ************************************************************************* //
```

blockMeshDict

```
    /*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  dev
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

convertToMeters 1;

halfAngle #calc "degToRad(2.5)";
c25 #calc "cos($halfAngle)*25";
s25 #calc "sin($halfAngle)*25";
c15 #calc "cos($halfAngle)*15";
s15 #calc "sin($halfAngle)*15";
c3 #calc "cos($halfAngle)*3";
s3 #calc "sin($halfAngle)*3";
c075 #calc "cos($halfAngle)*0.75";
s075 #calc "sin($halfAngle)*0.75";

ns25 #calc "sin(-$halfAngle)*25";
ns15 #calc "sin(-$halfAngle)*15";
ns3 #calc "sin(-$halfAngle)*3";
ns075 #calc "sin(-$halfAngle)*0.75";

// Element grading parameters
RefPara 0.9;
```

```
X1 #calc "$RefPara*40";
Z1 #calc "$RefPara*150";

X2 #calc "$RefPara*260";
Z2 #calc "$RefPara*150";

X3 #calc "$RefPara*260";
Z3 #calc "$RefPara*50";

X4 #calc "$RefPara*20";
Z4 #calc "$RefPara*120";

X5 #calc "$RefPara*260";
Z5 #calc "$RefPara*120";

vertices
(
        // Main control volume
        (30 0 0)
        (30 $s15 $c15)
        (30 $ns15 $c15)

        (0 0 0)
        (0 $s25 $c25)
        (0 $ns25 $c25)

        // GHV
        (11 0 0)
        (11 $s075 $c075)
        (11 $ns075 $c075)

        (10 0 0)
        (10 $s3 $c3)
        (10 $ns3 $c3)

        // Auxiliary points: refining surfaces
        (30 $s075 $c075)
        (30 $ns075 $c075)

        (11 $s25 $c25)
        (11 $ns25 $c25)

        (10 $s25 $c25)
```

```
        (10 $ns25 $c25)

        (0 $s3 $c3)
        (0 $ns3 $c3)
);

blocks
( // Ordered from lower Z to higher Z
        // Wake
        hex (3 18 19 3 9 10 11 9) ($X1 1 $Z1) simpleGrading (1 1 0.15)
        hex (18 4 5 19 10 16 17 11) ($X2 1 $Z2) simpleGrading (10 1 0.15)

        // Upper-body
        hex (10 16 17 11 7 14 15 8) ($X3 1 $Z3) simpleGrading (10 1 1)

        // Freestream
        hex (6 7 8 6 0 12 13 0) ($X4 1 $Z4) simpleGrading (1 1 20)
        hex (7 14 15 8 12 1 2 13) ($X5 1 $Z5) simpleGrading (10 1 20)
);

edges
(
        arc 2 1 (20 0 30)
        arc 15 14 (11 0 48.6667)
        arc 17 16 (10 0 50)
        arc 5 4 (0 0 50)
        arc 8 7 (11 0 0.75)
        arc 11 10 (10 0 3)

);

boundary
(
        axis
        {
                type empty;
                faces
                (
                        (6 0 0 6 )
                        (3 9 9 3)
                );
        }
    inlet
```

```
{
    type patch;
    faces
    (
                    (1 14 15 2)
                    (0 12 13 0)
                    (12 1 2 13)
    );
}

outlet
{
    type patch;
    faces
    (
                    (3 19 18 3)
                    (19 5 4 18)
    );
}

top
{
    type patch;
    faces
    (
                    (14 16 17 15)
                    (16 4 5 17)
    );
}

back
{
    type wedge;
    faces
    (
        (0 6 8 13)
                    (13 2 15 8)

                    (8 15 17 11)

                    (3 9 11 19)
                    (11 17 5 19)
    );
```

```
    }

    front
    {
        type wedge;
        faces
        (
            (0 6 7 12)
                        (12 7 14 1)

                        (7 10 16 14)

                        (9 3 18 10)
                        (10 18 4 16)
        );
    }


    ghv
    {
        type patch;
        faces
        (
                        (6 8 7 6)
                        (7 8 11 10)
                        (9 10 11 9)
        );
    }
);

mergePatchPairs
(
);

// ************************************************************************* //
```

controlDict

```
    /*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  dev
```

```
|
|    \\  /    A nd           | Web:      www.OpenFOAM.org
|
|     \\/     M anipulation  |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     rhoCentralFoam;

startFrom       startTime;

startTime       0;

stopAt          endTime;

endTime         0.12;

deltaT          1.75e-7;

writeControl    runTime;

writeInterval   0.0012;

purgeWrite      0;

writeFormat     ascii;

writePrecision  7;

writeCompression off;

timeFormat      general;

timePrecision   6;
```

```
runTimeModifiable true;

adjustTimeStep   yes;

maxCo            0.5;

maxAlphaCo       0.5;

maxDeltaT        1e-4;

functions
{
        #include "plotLiftDragMoment"
}


// ************************************************************* //
```

fvSchemes

```
    /*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  dev
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

fluxScheme      Kurganov;
```

```
ddtSchemes
{
    default          Euler;
}

gradSchemes
{
    default          Gauss linear;
        grad(p)          linearUpwind phi;
}

divSchemes
{
    default          none;
        div(phi,U)              Gauss vanLeerV grad(U);
        div(phi,h)              Gauss vanLeerV phi;
        div(u,rho)              Gauss vanLeerV phi;
        div(u,p)                Gauss vanLeerV phi;
}

laplacianSchemes
{
    default          none;
}

interpolationSchemes
{
    default          linear;
    reconstruct(rho) upwind phi;
    reconstruct(U)   upwind phi;
    reconstruct(T)   upwind phi;
}

snGradSchemes
{
    default          corrected;
}

// ************************************************************************* //
```

fvSolution

```
    /*---------------------------*- C++ -*----------------------------------*\
```

```
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  dev
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    "(rho|rhoU|rhoE)"
    {
        solver          diagonal;
    }

    U
    {
        solver          smoothSolver;
        smoother        GaussSeidel;
        nSweeps         2;
        tolerance       1e-10;
        relTol          0;
    }

    h
    {
        $U;
        tolerance       1e-10;
        relTol          0;
    }
```

```
}


// ************************************************************************* //
```

plotLiftDragMoment

```
    /*--------------------------------------------------------------------*/
/*
/* This fucntion calculates the aerodynamic forces and moments \\
of any case.
/*
/* Source: https://www.hpc.ntnu.no/display/hpc/OpenFOAM+ \\
-+Airfoil+Calculations
/* Just by specifing the parameters and the directions, \\
OpenFOAM has a library to calculate the coefficients
/*
/*--------------------------------------------------------------------*/

forces
{
    type                forces;
    functionObjectLibs  ("libforces.so");

    patches             (ghv);
    pName               p;
    UName               U;
    log                 true;

    CofR                (10.75 0 0);

    outputControl       timeStep;
    outputInterval      100;
}


forceCoeffs
{
    type                forceCoeffs;
    functionObjectLibs  ("libforces.so");

    patches             (ghv);
    p                   p;
    U                   U;
```

```
    rho             rhoInf;
    log             true;

    CofR            (10.75 0 0);
    liftDir         (0 0 -1);
    dragDir         (-1 0 0);
    pitchAxis       (0 -1 0);

    rhoInf          0.001;
    magUInf         1068.92;
    lRef            3;
    Aref            0.3927;

    outputControl   timeStep;
    outputInterval  100;
}
/*************************************************************/
```